STOCHASTIC PROGRAMMING APPROACHES FOR THE PLACEMENT OF GAS DETECTORS IN PROCESS FACILITIES

A Dissertation

by

SEAN DAVID WILEY LEGG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Carl D. Laird
Committee Members,	M. Sam Mannan
	Charles Glover
	Guy Curry
Department Head,	M. Nazmul Karim

August 2013

Major Subject: Chemical Engineering

ABSTRACT

The release of flammable and toxic chemicals in petrochemical facilities is a major concern when designing modern process safety systems. While the proper selection of the necessary types of gas detectors needed is important, appropriate placement of these detectors is required in order to have a well functioning gas detection system. However, the uncertainty in leak locations, gas composition, process and weather conditions, and process geometries must all be considered when attempting to determine the appropriate number and placement of the gas detectors. Because traditional approaches are typically based on heuristics, there exists the need to develop more rigorous optimization based approaches to handling this problem. This work presents several mixed-integer programming formulations to address this need.

First, a general mixed-integer linear programming problem is presented. This formulation takes advantage of precomputed computational fluid dynamics (CFD) simulations to determine a gas detector placement that minimizes the expected detection time across all scenarios. An extension to this formulation is added that considers the overall coverage in a facility in order to improve the detector placement when enough scenarios may not be available. Additionally, a formulation considering the Conditional-Value-at-Risk is also presented. This formulation provides some control over the shape of the tail of the distribution, not only minimizing the expected detection time across all scenarios, but also improving the tail behavior.

In addition to improved formulations, procedures are introduced to determine confidence in the placement generated and to determine if enough scenarios have been used in determining the gas detector placement. First, a procedure is introduced to analyze the performance of the proposed gas detector placement in the

face of "unforeseen" scenarios, or scenarios that were not necessarily included in the original formulation. Additionally, a procedure for determine the confidence interval on the optimality gap between a placement generated with a sample of scenarios and its estimated performance on the entire uncertainty space. Finally, a method for determining if enough scenarios have been used and how much additional benefit is expected by adding more scenarios to the optimization is proposed.

Results are presented for each of the formulations and methods presented using three data sets from an actual process facility. The use of an off-the-shelf toolkit for the placement of detectors in municipal water networks from the EPA, known as TEVA-SPOT, is explored. Because this toolkit was not designed for placing gas detectors, some adaptation of the files is necessary, and the procedure for doing so is presented.

To my family

ACKNOWLEDGEMENTS

I would like to sincerely thank my family for all of their support. Without their love, challenge, and support I would not be where I am today. My parents constantly set high goals and provided me the support and assistance needed to reach those goals and to set new ones. Thank you for encouraging me to attend graduate school, and keeping me motivated and encouraged through the entire process.

I would also like to thank my advisor Dr. Carl Laird. He has been critical in driving my success in graduate school. He has shown extreme patience and a tremendous gift for educating. His ability to identify my strengths and my weaknesses and develop strategies to use both has been absolutely crucial to my education. Dr. Laird was also a valuable friend and mentor, even beyond my academic career. I thank him for being a fantastic advisor and a great friend, and am absolutely grateful for having him as an advisor.

I would like to recognize the support I received from Dr. Sam Mannan and the Mary Kay O'Connor Process Safety Center. The exposure to new topics and to industry has been extremely important in my development.

I would like to thank my committee members Dr. Charles Glover and Dr. Guy Curry. Thank you for your advice and for serving on my committee the past several years.

I must also acknowledge Dr. Jean-Paul Watson and Dr. John Siirola of Sandia National Laboratories for their guidance and assistance in my research. They have provided many important insights and assistance in my work and in my publications.

I would like to thank my groupmates as well. My group is composed of a fantastic set of people to work with. I would specifically like to thank Daniel Word, Gabe

Hackebeil, Dr. Angelica Wong, and Alberto Benavides-Serrano. You have all acted as friends, collaborators, mentors, and even roommates to me at some point during my time in graduate school. Thank you all for everything you have done.

Finally, I would like to thank my fiancée and future wife, Monica. You have been supportive and understanding through long nights, stressful weekends, and long trips away. I am so lucky to have you in my life and I love you so much.

TABLE OF CONTENTS

Page

Page

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

1. INTRODUCTION: MOTIVATING THE USE OF OPTIMIZATION FOR SENSOR PLACEMENT*

Gas detection, specifically the detection of combustible and toxic gas release events, is a key component of modern process safety. Usually, combustible gas detectors are calibrated to alarm at a specified fraction of the lower explosive or lower flammable limit. These limits refer to the gas concentrations at which a dispersed gas cloud in air will allow a flame front to spread when exposed to an ignition source. Toxic gas detectors are designed to detect and alarm at a concentration related to the impact on human health, typically expressed in parts per million (ppm).

In addition to proper selection of the sensor specifications needed for a particular process facility, appropriate placement of these sensors is required to ensure effective detection of hazardous gas release events. There is significant uncertainty to consider when trying to determine the appropriate number and placement of gas detectors, including: leak location, gas composition, process conditions, the impact of surrounding geometries on dispersion, and weather conditions. Traditional approaches for gas detector placement are heuristic, typically based on source monitoring and providing a particular volumetric or linear coverage. Despite the fact that the importance of dispersion studies is generally recognized, the full value of these studies is often not utilized when determining a final gas detector placement.

In recent years, some new techniques have been developed to provide a more

*Part of this section is reprinted with permission from "A Stochastic Programming Approach For Gas Detector Placement Using CFD-based Dispersion Simulations" by Legg SW, Benavides-Serrano AJ, Siirola JD, Watson JP, Davis SG, Bratteteig A, and Laird CD, 2012. Computers & Chemical Engineering, 47(0):194-201, Copyright 2012 by Elsevier Ltd.

Part of this section is reprinted with permission from "Optimal Gas Detector Placement Under Uncertainty Considering Conditional-Value-at-Risk" by Legg SW, Wang C, Benavides-Serrano AJ, and Laird CD, 2013. Journal of Loss Prevention in the Process Industries, 26(3):410-417, Copyright 2013 by Elsevier Ltd.

quantitative treatment of the problem; however, there is still a need for a systematic approach that addresses the following issues:

- Key variables such as the leak location, weather conditions, process conditions, gas properties, and process geometries have a significant impact on gas dispersion. Rigourous simulation of this dispersion is necessary to accurately calculate important outputs (e.g., point concentrations, detection time, and cloud size) and, hence, the estimated performance of a particular gas detector layout. A thorough review of the influence of key variables and inputs is presented in the work by Kelsey et al. (2002), Kelsey et al. (2005), Bratteteig et al. (2011), and Marx and Cornwell (2009).

- Many of the existing practices are based on heuristics or analyses using a small number of high-impact scenarios. However, there is significant uncertainty in leak characteristics and weather conditions. There is a need for methods that provide rigorous treatment of these uncertainties. Unfortunately, given the large uncertainty space and the complexity of gas dispersion models, this requires plant-specific simulation of a large number of leak scenarios. However, given these scenarios, it is possible to calculate statistics of key performance metrics (e.g., expected value of the detection time).

- Most of the risk-based optimization techniques described above provide no guarantee of global optimality. Those that provide guarantees against suboptimality are based on general sensor placement metrics that do not consider key variables of importance to gas detector placement in process facilities. There is a need for an optimization-based approach that rigorously considers uncertainties and provides guarantees of optimality while making use of the valuable plant-specific information provided from rigorous gas dispersion simulations.

In this research, we seek to answer several key questions. First, can we develop a mathematical formulation for the placement of gas detectors that is both realistic and computationally tractable? Second, can we define an appropriate objective function for this problem? Can we perform enough simulated release scenarios to accurately quantify the associated risk? Furthermore, can we apply an appropriate set of constraints that allow for unknown scenarios to be accounted for? Finally, when a gas detector placement has been determined, how can we know whether enough simulations have been utilized or if the proposed placement is satisfactory for the entire uncertainty space?

We present a stochastic programming formulation for determining the optimal plant-specific placement of gas detectors. The uncertainty associated with different leak scenarios is captured through hundreds of process-specific computational fluid dynamics (CFD) simulations using FLACS, a package for rigorous modeling and simulation of explosions and gas dispersion. Using simulation results from the dispersion model, a multi-scenario, mixed-integer linear programming (MILP) formulation is developed to perform optimal placement of gas detectors by minimizing the expected detection time over all scenarios.

A second formulation is developed that includes a constraint on the overall coverage provided by the detector placement. While this formulation produces a slightly poorer objective value when optimizing over a particular subsample of scenarios, it is more resilient to release scenarios not considered by the optimization problem. The use of a coverage constraint reflects many of the heuristics and sensor placement tactics already in place through the process industry.

Each of the previous formulations seek to minimize the expected detection time across all scenarios. However, it may often be beneficial to optimize other objective functions. While a low expected detection time is desirable, this does not necessarily

minimize the "worst-case" scenarios, or those that require the most time to detect or provide the worst results. Therefore, a formulation that seeks to minimize the Conditional-Vale-at-Risk is also developed. This formulation is designed to provide control over the tail distribution of scenarios in the full scenario space, minimizing the detection time of the scenarios that require the most time, while also only suffering a minor penalty in the expected detection time across all scenarios.

While the time required to solve the resulting MILP is small (on the order of a few CPU seconds), the computational bottleneck is simulation of the leak scenarios (on the order of CPU hours or days per scenario). Given the high computational cost, the number of scenarios available for optimization may not be sufficient to adequately represent the uncertainty space. This can be especially problematic in sensor placement problems if the number of scenarios is not significantly larger than the number of sensors to be placed. Approaches based purely on the minimization of the expected impact may find solutions that protect against the specific scenarios considered in the optimization, but provide little general robustness against alternate scenarios. Therefore, we use the procedure described in Mak et al. (1999) to determine confidence intervals on the value of the objective function and quantify the effectiveness of candidate sensor placements on scenarios not considered in the optimization.

Additionally, when determining sensor placements through the methods proposed in this paper, it is important to determine how any two proposed sensor placements may vary. Because different scenario sets can yield different sensor placements, it is important to quantify the difference between any two proposed gas detector placements. Metrics for comparing seemingly different sensor placements are proposed and examined. This technique is tested on a similar problem to the gas detector problem, the placement of sensors in a municipal water network. This problem is

ideal because it allows for a scenario set to be generated that covers the entire uncertainty space, a task that is impossible for the gas dispersion problem, thus making it easy to assess the efficacy of the method.

Finally, the use of an already available toolset from the EPA, TEVA-SPOT (Hart et al., 2008), is discussed. TEVA-SPOT was originally designed as a tool for municipal water network managers to develop sensor placements for the prevention of water contamination. The necessary changes to the gas dispersion data are discussed, as well as how to utilize this software to generate a gas detector placement for a process facility.

This document is organized in the following manner. The remainder of this chapter covers a literature review of the common, current industrial practices for gas detector placement, the previous work in detector placement and facility location optimization, an overview of the data set used for the results in this document, and the hardware and software used to generate all results. Chapter 2 presents the multi-scenario, mixed-integer linear programming formulation for optimal placement of gas detectors and numerical results. Chapter 3 presents a modification to the original problem formulation that incorporates constraints on minimum coverage distances. Chapter 4 introduces the CVaR formulation, along with numerical results. In Chapter 5, the procedure of Mak et al. (1999) is introduced and utilized to provide confidence intervals on solutions generated by the MILP formulation. Solution variability is discussed in Chapter 6. Finally, the use of the off-the-shelf tool TEVA-SPOT is covered in Chapter 7.

1.1 Current Industry Practices for Sensor Placement

1.1.1 Gas Detector Types and Properties

Three types of sensors are commonly used for the detection of combustible gas clouds: catalytic point detectors, infrared point and beam detectors, and ultrasonic area detectors (Nolan, 2010; UK Health & Safety Executive, 2011a). Originally developed in 1958 for the mining industry, catalytic point gas detectors are the most widely used gas detector in the process industry. Catalytic gas sensors detect the presence of a chemical contaminant by an oxidation reduction reaction with the catalyst. The heat generated from the oxidation reaction on the detector's catalyst provides an accurate reading of the concentration of contaminant in the area. An example of the bead catalyst used in these detectors can be seen in Figure 1.1 from Delphian Detection Technology (2013). These detectors are used to detect all combustible gases, though may not necessarily have an equal reaction to different gases. In addition to providing accurate concentration measurements for the released gases, these detectors have a very low unit cost. This low unit cost makes them an attractive sensor for use throughout a facility. However, there are three major drawbacks of the catalytic gas detectors. First, these detectors are point detectors only. This requires that the contaminant cloud pass directly through the area of the gas detector. If the weather or ventilation patterns are not properly analyzed, these detectors can be placed poorly and therefore be rendered ineffective. Because these detectors require a catalyst to detect contaminant gases, they are subject to catalyst poisoning by a variety of materials in air. These poisons include tetraethyl lead, glycols, flame inhibitors in plastics, sulfur compounds, dirt, and several other chemicals that may be found in petrochemical facilities. All of these can impair detector reliability, or even cause full failure of the detector, both of which are extremely difficult to detect

without regular maintenance as the detectors fail in a false negative manner. By failing in a false negative, the sensor will act as if there is no contaminant in the area, even if a leak may exist.

Figure 1.1: Catalytic bead sensor from a gas detector manufactured by Delphian Detection Technologies.

The second major type of detector is the infrared (IR) gas detector. These detectors can be designed to provide two different types of coverage: point and beam detection. The point detector works in a similar fashion to the catalytic point detector; a contaminant must pass directly across the detector and an analyzer determines the presence of the gas. Gas is trapped in the detector's sample cell, and an infrared beam is passed through the cell (Chou, 2000). If a gas is present, some of this infrared energy will be absorbed. The absorption of this energy is dependent on the type of gas present, as each gas has an individual "fingerprint" based on the atoms

and bonds present in the molecules. These detectors can detect the gas through two mechanisms. Because each gas molecule has a particular absorption frequency, it will vibrate more rapidly when the infrared beam passes through it, causing the gas to heat up. The detector can recognize this rise in temperature and signal that a contaminant is present. Additionally, some of the infrared energy will be absorbed, so the detector can recognize that less energy was received than emitted which also indicates the presence of a contaminant. Because the contaminant does not need to interact directly with a catalyst, as all of the detectors are optical, removing the possibility of catalyst poisoning causing a sensor failure. Unfortunately, the IR point gas detectors also suffer from the same location dependent effectiveness as the catalytic point detector. Recently, the IR beam detector has been introduced to the industry. An example of this type of detector can be seen in Figure 1.2 from Det-Tronics (2013). This detector operates along the same infrared absorption principle as the point detector, but does so over a long distance. An IR emitter is place on one end of the beam, while a receiver is place at the other end. When a significant difference between the radiation energy emitted and received is detected, the sensor considers a contaminant detected (UK Health & Safety Executive, 2011b). These sensors possess a higher unit cost and lower overall accuracy in terms of concentration quantification, but can often detect contaminant over a large area and require less maintenance. While these sensors are not as effective at precisely determining the concentration of a contaminant cloud, they can detect releases over larger distances. False positives may be triggered by steam or fog, which is the major drawback to these types of detectors.

An additional option for detecting gas releases over large distances is the acoustic gas detector. These detectors do not operate by detecting the contaminant itself, but by detecting the release of the contaminant (**?**). As a gas escapes through a

Figure 1.2: Open-path infrared gas detector from Det-Tronics.

hole from a high pressure to a low pressure, the rapid of expansion causes turbulent flow, resulting in an audible sound. This sound lies in the 20 Hz to 10 MHz range, spanning both the audible and ultrasonic sound ranges. The gas detectors specialize in detecting the ultrasonic sound wave, even for very low leak rate, often as low as 0.1 kg/s. By detecting the leak rate, the gas cloud does not need to accumulate or spread in order to be detected, thus resulting in a much more immediate detection. Each of these detectors may cover an area up to 20 m in radius from the actual detector, and new designs have self-testing mechanisms. Small microphones will emit ultrasonic sounds to verify the detectors are working, reducing the amount of human-involved maintenance necessary. An example of an acoustic detector produced by Gassonic A/S is shown in Figure 1.3 (**?**). Unfortunately, if the gas releases occurs at a slow rate, or from a large enough hole that the gas flows in a less than turbulent manner, these detectors will be rendered ineffective. Also, some normally occurring process noises may falsely trigger these detectors.

For toxic release events, electrochemical or semiconductor gas detectors are typically used. Electrochemical detectors allow a gas to diffuse through a porous mem-

Figure 1.3: Acoustic gas detector from Gassonic A/S.

brane to an electrode where it can either be oxidized or reduced to determine concentration. A semiconductor sensor detects a gas by observing changes in resistance on a conductive surface in the presence of a contaminant. Both detector types are calibrated to raise an alarm at a chemical-specific threshold concentration.

All of the sensors mentioned can be used in either a fixed or portable form. This paper considers fixed sensors only.

1.1.2 Industrial Standards for Sensor Placement

The placement of gas detection sensors in the petrochemical industry, whether for toxic or flammable releases, is typically based upon prescriptive processes and rules-of-thumb that rely on identifying equipment of interest, credible scenarios, and the properties of the gas in question. Following guidelines in the CCPS (2009) classification, the qualitative methods for the placement of gas detectors can be categorized according to their main intended purpose: source monitoring, volumetric monitoring, enclosure monitoring, path of travel and target receptor monitoring, and perimeter monitoring. Particular examples where these approaches are encouraged can be found in NFPA 15 (Section 6.5.2.7.1) (NFPA, 2007), API RP 14C (Section

C.1.3.2) (API, 2001), ANSI/ISA-RP12.13.02 (IEC 61779-6 Mod) (ANSI, 2003), IEC 60079-29-2 (IEC, 2007), HSE (Section 6) (HSE, 1993), HSL (Section 4.6) (HSL, 2001), UKOOA (1995), Nolan (1996) (Section 17.6), and ISA-TR84.00.07 (Annex A.2, Step 7) (ISA, 2010).

1.1.3 Risk-based Approaches to Sensor Placement

In recent years, the aforementioned approaches have been coupled with risk concepts in order to enhance their effectiveness. In ISA-TR84.00.07 (ISA, 2010), a mitigated risk assessment is performed based on gas detector coverage and, if the desired risk threshold is not met, the gas detector placement is modified. In DeFriend et al. (2008), a risk-based methodology is proposed to assess the maximum size of a gas cloud that must be detected for the facility to have a tolerable risk. Placement methodologies based on optimization concepts have been considered as well. Obenschain et al. (2004) proposed the use of a genetic algorithm based on elitist selection where the members of the population with the highest fitness scores are kept in the next population. In Lee and Kulesz (2008), sensor locations are determined via an iterative dynamic programming algorithm using a risk measure as the objective function. In Gencer et al. (2008), the proposed mathematical models make it possible to optimally locate chemical agent detectors and alarms to provide the best interaction between the two systems. Dhillon and Chakrabarty (2003) proposed two general purpose algorithms based on miss probability and coverage considerations. Strøm and Bakke (1999) proposed a performance-based algorithm for the placement where a matrix of detectors is defined within the volume and an overall efficiency value for each specific detector is calculated. Sensor locations are then selected based on their ranking according to this metric. These approaches show promise over qualitative techniques; however, there is still a need for a much more systematic approach to

the placement of gas detectors.

1.2 Literature Review of Optimization Techniques for Sensor Placement

The stochastic programming formulation presented in this paper addresses the concerns that the information provided by gas dispersion simulations is not properly being utilized. Hundreds of scenarios across different leak locations and weather conditions are generated by rigorous simulation of the gas dispersion using FLACS. Writing the extensive form of the stochastic program results in a mixed-integer linear programming formulation that can handle a large numbers of scenarios, and is solvable to global optimality with little computational effort.

1.2.1 Sensor Placement in Municipal Water Networks

The work presented in this paper is an extension of an optimization approach developed by Berry et al. (2004) for the placement of contamination sensors in large-scale water distribution networks. By precomputing a large set of possible contamination scenarios, point sensors are placed throughout the network in order to minimize the total consumption of contaminant across a network subject to a set of cost constraints. Extensions to this work were made to relax constraints that required each event to be detected by at least one sensor with the addition of an unphysical dummy location (Berry et al., 2005). Further extensions considered effects of time on the sensor placement (including detection time) (Berry et al., 2006b) and uncertainty in flow patterns (Shastri and Diwekar, 2006). Early work in sensor placement for the detection of accidental contaminations in municipal water networks was performed by Kessler et al. (1998). Ostfeld and Salomons (2004) extended this problem to include unsteady conditions and applied a genetic algorithm to maximize coverage.

Watson et al. (2004) analyze the benefits of and tradeoffs between different objective functions. Berry et al. (2009) develop a formulation that accounts for imperfect

sensors with probability of missing a detection. Robust formulations consider solving an optimization problem in which the effects of the worst-case scenario in a set of scenarios is minimized (Carr et al., 2006; Watson et al., 2009).

Berry et al. (2005) assume that point sensors detect contamination if and only if the contaminant physically passes through the sensor. The sensors used in their formulation exhibit the same properties possessed by infrared and catalytic point gas sensors used in the petrochemical industry (Fire & Safety World Online, 2011). The water sensor network problem and the open-air gas dispersion problem differ mainly in the simulation frameworks used to create the possible contamination (or gas leak) scenarios. An adaptation of the sensor placement formulation in Berry et al. (2005) was applied to the open-air problem for homeland security purposes by Hamel et al. (2006). In their work, a CFD model is used to generate a set of potential urban attack events where a nuclear, biological, or chemical agent is dispersed into air.

CFD modeling of the dispersion is analogous to the EPANET water network model used in the work by Berry et al. (2006b). Our work considers flammable or toxic gas releases from a chemical process facility, and CFD software is used to generate rigorous dispersion simulations for various leak locations, process conditions, and weather variables.

1.2.2 p-Median Facility Location Problems

The mixed-integer linear programming formulation we use for the placement of sensors is identical to the p-median facility location problem (Hakimi, 1965; ReVelle and Swain, 1970; Mirchandani and Francis, 1990). The p-median problem can be solved with a branch-and-cut MILP solver in cases where problem sizes are not too large (Berry et al., 2006b), such as those presented in this paper. In the event that the problem size becomes too large, numerous heuristics exist for solution of these

problems. A hybrid heuristic for solving the p-median problem has been developed by Resende and Werneck (2004) which utilizes a Greedy Randomized Adaptive Search Procedure (GRASP) to generate a set of solutions, followed by local search and path-relinking to obtain high-quality local optima. Additionally, Berman et al. (2007) extend the p-median problem to consider facility reliability and disruptions. This idea is analogous to sensor failure in the sensor placement formulation and has been addressed by Benavides-Serrano et al. (2012).

1.3 CFD-Dispersion Data

The leak scenarios for this paper were generated by GexCon using the CFD package, FLACS (GexCon, 2011). Three separate sets of data are used in this paper to demonstrate and investigate the sensor placement techniques presented. A set of leak scenarios were generated for a real process facility, varying the leak locations and weather conditions. There were 270 scenarios generated for data set 1, 314 scenarios for data set 2, and 145 scenarios for data set 3. Each data set represents a separate module located within the same petrochemical processing facility. The process geometry itself is proprietary, however, it represents the full process geometry (equipment, piping, support structures, etc.) for a facility that is approximately 20 m in height and 50 m by 70 m in area for data set 1. The size of the process facility for data set 3 is of similar size, while the facility for data set 2 is a bit smaller (20 m in height and 20 m by 35 m in area). All scenarios were considered to have equal probability, thus the probability term α_a in the formulations is the same for all scenarios $a \in A$. A total of 994 potential point sensor locations were considered for data set 1, while 768 potential sensor point locations were considered for data set 2 and 943 potential locations for data set 3. Using data from each leak scenario simulation, GexCon identified the list of sensor locations where the concentration

of the gas was more than 10% of the LFL value, signaling that detection of the leak scenario would be possible from those locations. In addition, they provided the detection times $d_{a,i}$ for each of the scenarios $a \in A$ and locations $i \in \mathcal{L}_a$. Given this data, the performance metric selected for these case studies was the expected value of the detection time. Of course, other measures, such as the volume of the flammable cloud at the time of detection, could also be used depending on available data.

1.3.1 Governing Physics in FLACS

The CFD package FLACS solves the compressible fluid flow conservation equations on a 3D Cartesian grid via the finite volume method. The conservation equations include those for mass, momentum, and species mass fractions as well as relevant transport equations. The basic equations used in FLACS are described by Hjertager (1984, 1986) and can be found in the FLACS user's manual (GexCon, 2010).

1.3.1.1 Governing Equations for Fluid Flow

A mathematical model for compressible fluid flow is included in FLACS. The conservation of mass is accounted for, as is the momentum equation. Transport equations for enthalpy, mass fraction of species, and kinetic energy (and the dispersion thereof) are also included. The production of turbulent kinetic energy is contributed to by flow shear stresses, wall shear stresses, buoyancy, and sub-grid objects.

1.3.1.2 Turbulence Model

In FLACS, turbulence is modelled through the $k-\varepsilon$ model. This model is an eddy viscosity model that solves two of the transport equations mentioned above,

the turbulent kinetic energy and the dissipation of turbulent kinetic energy. A set of Prandtl-Schmidt numbers describing the diffusion of a variable compared to the dynamic viscosity are defined by FLACS.

1.3.1.3 Boundary Layers

In regions nearest to walls and obstructions, steep gradients in turbulent kinetic energy and its dissipation exist. These regions are known as boundary layers, and the influence of these walls are modelled in FLACS. All energy transport equations are defined in this boundary region. Additionally, boundaries for the atmospheric layers are also defined. These wind boundaries are dependent on the Pasquill stability classes and the Monin-Obukhov length. The Pasquill stability classes are shown in Table 1.1, and show how this boundary layer height is affected by the Pasquill class. The value L is the Monin-Obukov length, u is the wind velocity, and f is a turbulence generation factor.

Pasquill Class	Stability	Boundary Layer Height, h
A	Unstable	1500 m
B	Unstable	1500 m
C	Slightly Unstable	1000 m
D	Neutral	$0.3u^* \frac{L}{f}$ m
E	Slightly Stable	$0.4\sqrt{\frac{u^* L}{f}}$ m
F	Stable	$0.4\sqrt{\frac{u^* L}{f}}$ m

Table 1.1: Wind Profile Parameters

1.4 Software and Hardware

The problem formulations were all formulated in Pyomo and solved using CPLEX 12.2 (IBM, 2010) on a dual quad-core Intel(R) Xeon(R) CPU X5482 with a clock speed of 3.2GHz and 18 GB RAM.

The mixed-integer problems presented in this paper were formulated and solved using Pyomo (Hart et al., 2011). The Python Optimization Modeling Objects (Pyomo) software package provides a complete optimization modeling platform in a Python environment that allows for full scripting capabilities. Included in the Pyomo package are classes for defining parameters, variables, sparse sets, objective functions, and constraints. Pyomo can be used to formulate and solve linear, mixed-integer linear, nonlinear, or nonlinear mixed integer models. The Pyomo package is part of the Coopr (COmmon Optimization Python Repository) software library (COOPR, 2009). Coopr includes interfaces to common linear, mixed-integer, and nonlinear solvers, allowing users to apply optimizers to models developed using Pyomo.

2. GENERAL MIXED-INTEGER FORMULATION FOR GAS DETECTOR PLACEMENT*

In this section, a stochastic programming formulation for determining the optimal placement of gas detectors in petrochemical facilities, referred to as Problem (SP), is presented. Our goal is to determine a gas detector placement that is optimal (in some statistical metric) while considering uncertainty in the leak scenario by performing a single optimization over a large ensemble of potential leak scenarios. Variations in environmental conditions, process conditions, process geometries, and gas properties yields this large uncertainty space. The uncertainty is captured here through hundreds of computational fluid dynamics (CFD) simulations that are develop for the specific facility being considered. A multi-scenario, mixed-integer programming formulation utilizes these simulations to determine an optimal gas detector placement by minimizing the detection time over all scenarios subject to a set of constraints. These constraints limit the maximum number of allowable sensors and require that all scenarios be detected by at least one sensor.

In the next section, the MILP formulation for optimally locating gas detectors throughout a facility is presented. In Section 2.2, this formulation is solved for a particular facility using the CFD simulations provided by GexCon. These results show how the fraction of set of events detected and the expected detection time across all events is impacted by the number of sensors allowed. Finally, Section 2.3 provides a summary of the chapter.

*Part of this section is reprinted with permission from "A Stochastic Programming Approach For Gas Detector Placement Using CFD-based Dispersion Simulations" by Legg SW, Benavides-Serrano AJ, Siirola JD, Watson JP, Davis SG, Bratteteig A, and Laird CD, 2012. Computers & Chemical Engineering, 47(0):194-201, Copyright 2012 by Elsevier Ltd.

2.1 MILP Formulation

Problem (SP), the mixed-integer linear programming formulation for optimal gas detector placement is given by,

$$\min \quad \sum_{a \in A} \alpha_a \sum_{i \in \mathcal{L}_a} d_{a,i} x_{a,i} \tag{2.1a}$$

s.t.

$$\sum_{l \in L} s_l \leq p \tag{2.1b}$$

$$x_{a,i} \leq s_i \qquad \forall\, a \in A,\, i \in \mathcal{L}_a \tag{2.1c}$$

$$\sum_{i \in \mathcal{L}_a} x_{a,i} = 1 \qquad \forall\, a \in A \tag{2.1d}$$

$$s_l \in \{0, 1\} \qquad \forall\, l \in L \tag{2.1e}$$

$$0 \leq x_{a,i} \leq 1 \qquad \forall\, a \in A,\, i \in \mathcal{L}_a. \tag{2.1f}$$

The notation used in the problem is summarized in the Nomenclature section of the preamble. The set L=$\{1, 2, \ldots N\}$ includes all the potential gas detector locations, and A=$\{1, 2, \ldots, M\}$ represents the set of leak scenarios considered. Not all sensor locations are affected by each leak scenario, and the subsets $\mathcal{L}_a \subseteq L,\ \forall\ a \in A$, are defined such that $\mathcal{L}_a$ contains all the sensor locations that can detect leak scenario a.

The objective function represents the expected value of the desired performance metric. The parameter α_a is the probability associated with leak scenario a, and the parameter $d_{a,i}$ is the applicable damage coefficient if leak scenario a is first detected by a sensor at location i. The damage coefficients $d_{a,i}$ are precomputed for each release scenario $a \in A$ and each sensor location $i \in \mathcal{L}_a$. For the case studies considered in this paper, each scenario is assumed equally probable, and we seek to minimize

the expected value of the detection time. Therefore, $d_{a,i}$ is the time required for a sensor at location i to detect the leak simulated in scenario a. In general, expert advice could be used to determine alternate values for α_a, and the precomputed damage coefficient can be any computable performance metric. For example, $d_{a,i}$ may alternatively be defined as the total volume of flammable gas from scenario a at the time corresponding to detection by a gas detector at location i. Similarly, in the event of a toxic gas release, it may be more appropriate to define $d_{a,i}$ as the volume of gas above some toxic threshold, such as the LC_{50}. Because it is precomputed, any performance metric can be selected, regardless of complexity or nonlinearity, while still retaining linearity of the optimization formulation.

The binary decision variable s_l represents the existence of a sensor (s_l=1) or the lack of a sensor (s_l=0) at location l. The constraint given by equation (2.1b) limits the maximum number of gas detectors placed to be no more than the parameter p. Each $x_{a,i}$ is an indicator variable that has a value of 1 if a sensor at location i is the first to detect leak scenario a, and 0 otherwise. Constraint (2.1c) ensures that location i can only be the first to detect leak scenario a if there is a sensor placed at location i.

While constraint (2.1d) requires that each leak scenario must be detected by at least one sensor, a dummy location is included in $\mathcal{L}_a$ for every scenario, representing the situation where scenario a is not detected (Berry et al., 2006b). The damage coefficients for these dummy locations correspond to the maximum possible damage should the scenario go undetected. The $x_{a,i}$ variables are continuous, however, given the problem formulation, each $x_{a,i}$ will solve to either 0 or 1 in a converged solution (Berry et al., 2006b).

2.2 Numerical Results

Given this data, problem (SP), defined by equations (2.1a-2.1f), represents a stochastic programming problem that solves for an optimal placement of gas detectors that minimizes the expected value of the detection time across the scenarios considered. Using this formulation, we first solve a sequence of problems with an increasing value for p, the maximum number of sensors. For each scenario, a dummy location is included to allow for the possibility that a release scenario is not detected. In these case studies, the damage coefficient for the dummy location is set to be 10 seconds greater than the largest damage coefficient provided by the CFD simulations. For data sets 1 and 3, this value is 510 seconds, while for data set 2 this value is 900 seconds. This value penalizes any leak scenarios that are undetected. Results will be shown for each of the three data sets.

2.2.1 Data Set 1 Results

Figure 2.1 shows the optimal results as a function of the number of sensors placed. The $\bullet$ symbols show the expected value of the detection time (indicated on the left ordinate). The $\times$ symbols show the fraction of scenarios that are detected (indicated on the right ordinate). The large damage coefficient assigned to the dummy location penalizes undetected scenarios, and all scenarios are detected when 24 or more gas detectors are allowed. This number of detectors is consistent with the results when detection is required by not allowing the use of a dummy location. After this point, additional sensors are not required to improve the number of leak scenarios detected, however they provide reduction in the expected detection time. Each of these optimization problems required only a few seconds to solve. Figure 2.1 was generated using problem (SP).

Figure 2.1 provides an important tool for safety systems planning and design. It

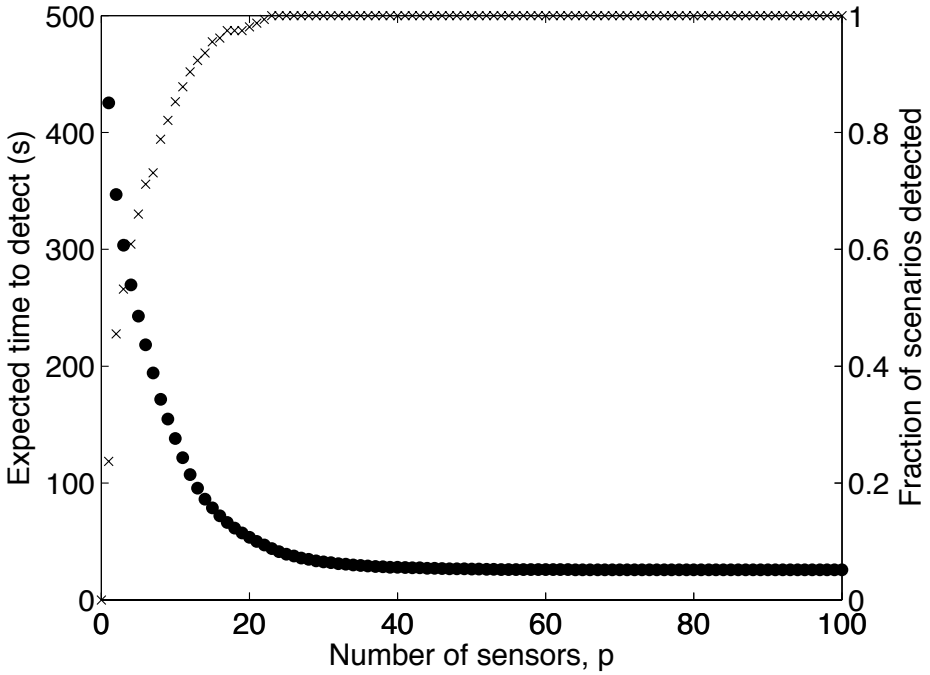

Figure 2.1: Objective vs. maximum allowable sensors for Data Set 1. Expected detection time (•) and fraction of leak scenarios detected (×) as a function of the maximum number of allowed sensors, p.

supplies a plant-specific quantification of the number of sensors against the expected value of the detection time (or other desired metric). Decision-makers can use this information to evaluate the tradeoff between the cost of this component of their safety system and its expected effectiveness.

Figure 2.2 shows the relative locations of the chosen sensors for our example facility in data set 1 with p, the maximum number of sensors, set to 50. The diagram represents an overhead view of the facility. The sensors may be located at differing vertical locations, which is not shown in this figure. The potential locations for the sensors are shown by the × markings, while the sensor locations chosen by Problem (SP) are shown by the □ markings. There are several areas, specifically toward the left side of the figure, where the are no sensors placed. In future chapters, we will

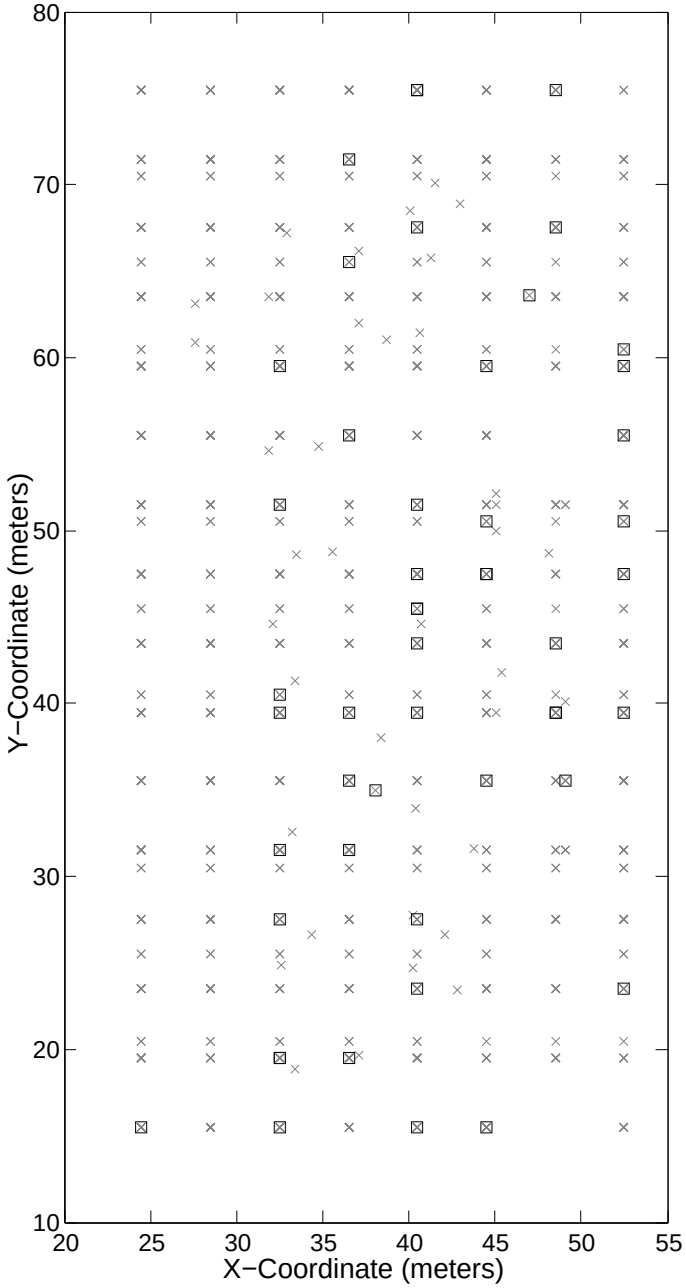

Figure 2.2: Locations of sensors placed by Problem (SP) in the process facility for Data Set 1. $\times$ represent potential sensor locations, $\square$ represent sensors placed by formulation (SP).

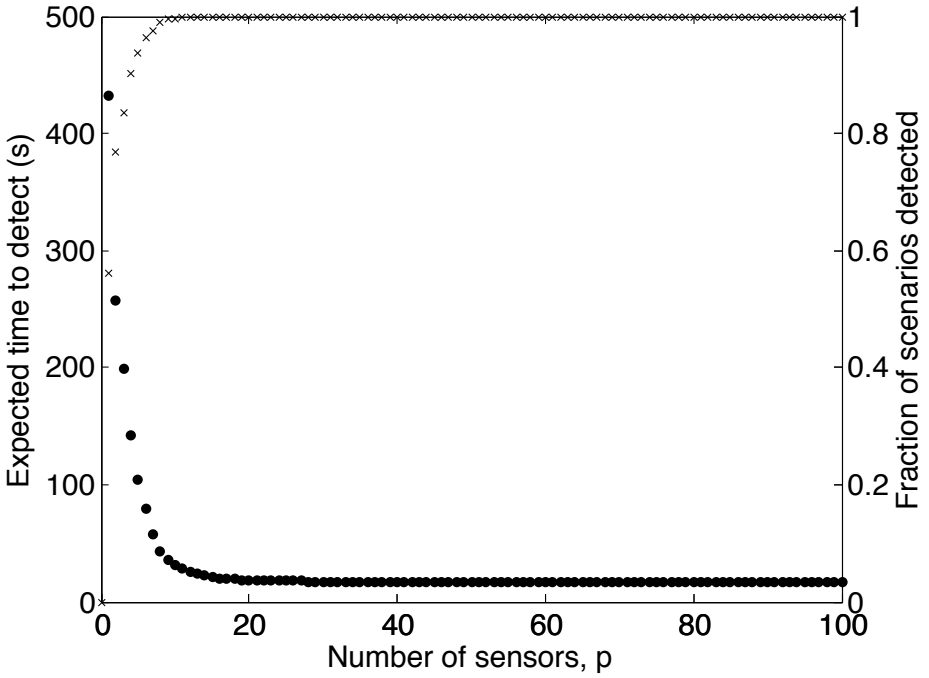

Figure 2.3: Objective vs. maximum allowable sensors for Data Set 2. Expected detection time (•) and fraction of leak scenarios detected (×) as a function of the maximum number of allowed sensors, p.

demonstrate how adjusting the problem formulation can improve coverage in these areas.

2.2.2 Data Set 2 Results

Figure 2.3 shows the results as a function of the number of sensors placed for data set 2. The • symbols show the expected value of the detection time (indicated on the left ordinate). The × symbols show the fraction of scenarios that are detected (indicated on the right ordinate). The large damage coefficient assigned to the dummy location penalizes undetected scenarios, and all scenarios are detected when 12 or more gas detectors are allowed. After this point, additional sensors are not required to improve the number of leak scenarios detected, however they do provide reduction

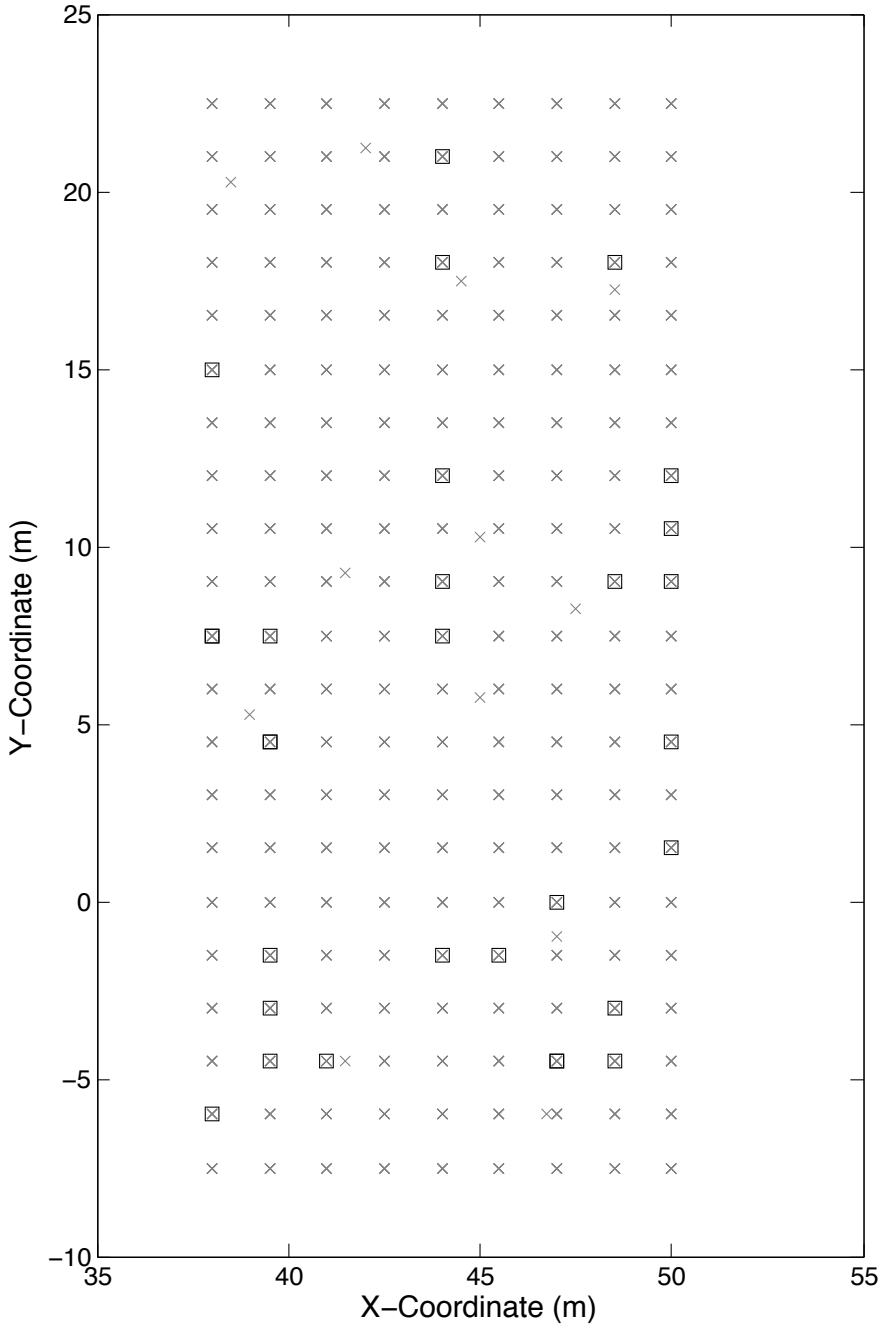

Figure 2.4: Locations of sensors placed by Problem (SP) in the process facility for Data Set 2. $\times$ represent potential sensor locations, $\square$ represent sensors placed by formulation (SP).

in the expected detection time. Again, each of these optimization problems required only a few seconds to solve. Figure 2.3 was generated using problem (SP).

A total of 30 sensors were placed throughout the facility, as shown in Figure 2.4. The locations of the sensors selected by Problem (SP) are shown by the $\square$ markings, while the potential locations are shown by the $\times$ markings. This layout is an overhead view, looking down on the facility. These sensors may be located at different vertical locations, though this is not captured in this figure. Several areas are devoid of sensors because there are no scenarios included in the data set that impact this area. In the next chapter, we will investigate the use of coverage constraints to locate sensors in these areas in order to improve the resiliency to unforeseen scenarios for Problem (SP).

2.2.3 Data Set 3 Results

In Figure 2.5, the expected detection time across all scenarios as a function of the number of sensors placed is shown for data set 3. The $\bullet$ symbols show the expected value of the detection time (indicated on the left ordinate). The $\times$ symbols show the fraction of scenarios that are detected (indicated on the right ordinate). The large damage coefficient assigned to the dummy location penalizes undetected scenarios, and all scenarios are detected when 18 or more gas detectors are allowed. After this point, additional sensors are not required to improve the number of leak scenarios detected, however they do provide reduction in the expected detection time. As with the previous two examples, each of these optimization problems required only a few seconds to solve. Figure 2.5 was generated using problem (SP).

A set of 55 sensors were placed throughout the facility, as shown in Figure 2.6. The locations of the sensors selected by Problem (SP) are shown by the $\square$ markings, while the potential locations are shown by the $\times$ markings. This layout is an overhead

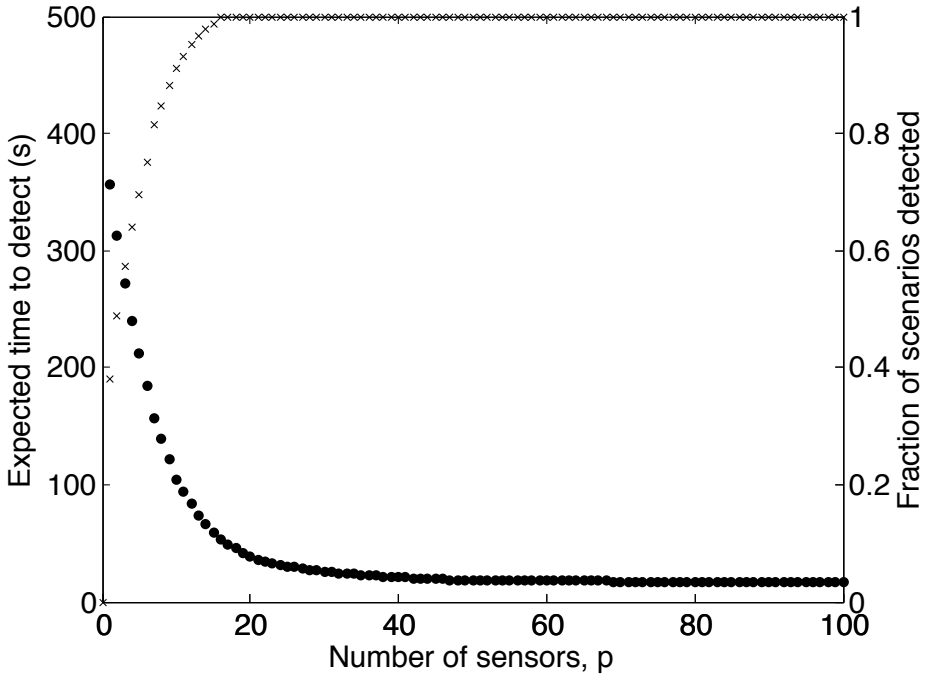

Figure 2.5: Objective vs. maximum allowable sensors for Data Set 3. Expected detection time (•) and fraction of leak scenarios detected (×) as a function of the maximum number of allowed sensors, p.

view, looking down on the facility. These sensors may be located at different vertical locations, though this is not captured in this figure.

2.3 Summary

This chapter presented a mixed-integer programming formulation for the optimal placement of gas detectors upon which the rest of this dissertation is built. This formulation allows seeks to optimally place a finite number of gas detectors, even in the face of a large uncertainty space in the set of potential gas release events. A large number CFD simulations are necessary to provide this leak scenario data, each with varying conditions to represent a different portion of the uncertainty space. Problem (SP) sought to optimally place a set of gas detectors in order to minimize the expected detection time across the entire set of release scenarios. The expected

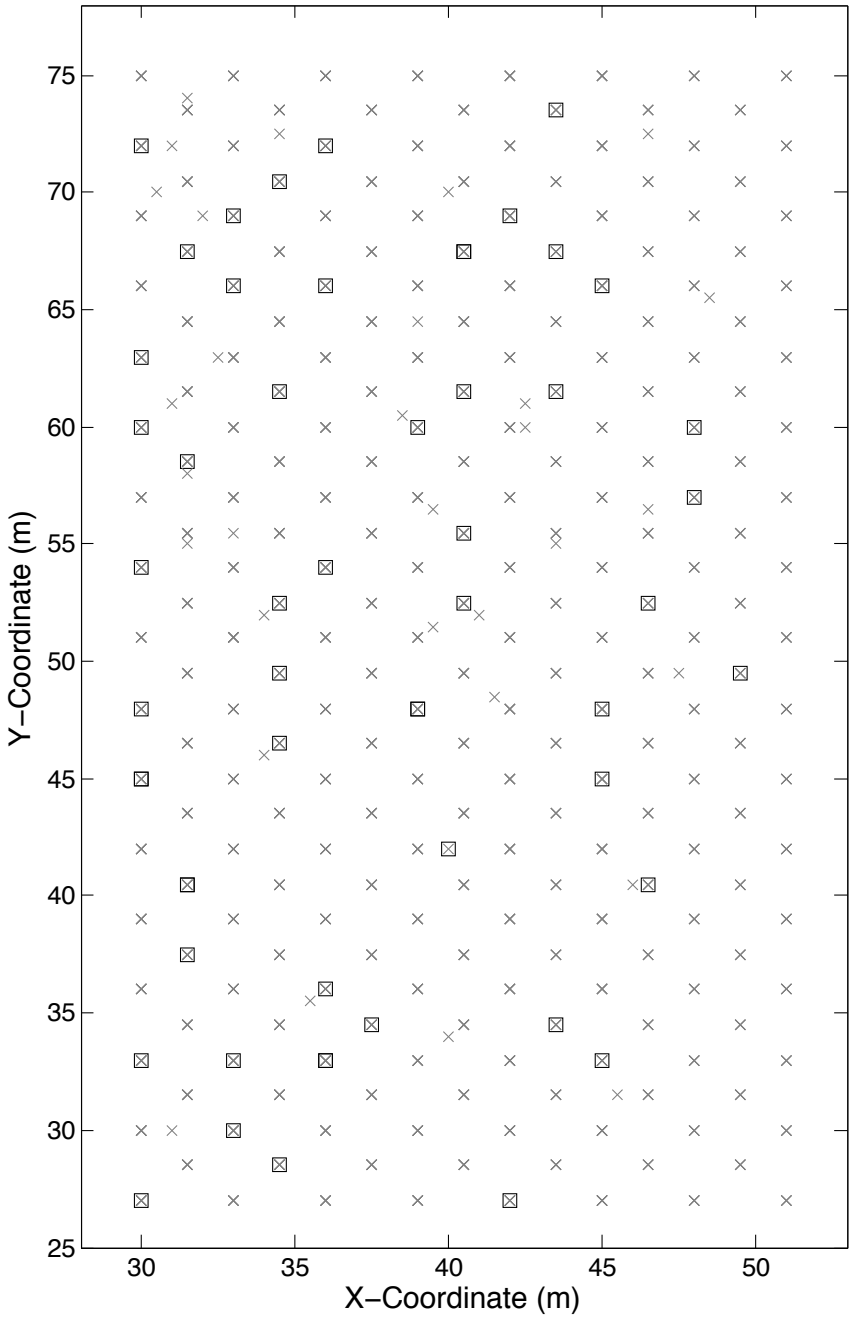

Figure 2.6: Locations of sensors placed by Problem (SP) in the process facility for Data Set 3. $\times$ represent potential sensor locations, $\boxtimes$ represent sensors placed by formulation (SP).

detection time was the metric utilized here, though it should be noted that additional metrics such as expected cloud size, expected impact of an explosion, or expected number of impacted individuals could be utilized if that data is available without modifying the problem formulation. The problem formulation was implemented in the Pyomo optimization modeling framework.

Numerical results for this formulation were provided using the data sets representing real process geometries. This placement is provably optimal with respect to the scenarios used to build the extensive form of the stochastic program. The formulations were shown to work on three separate data sets, providing usefulness for multiple process geometries. Solution of this mixed-integer programming problem was computationally efficient, requiring only a few seconds to determine an optimal gas detector placement. This computational efficiency is very promising, as it indicates that this technique for determining sensor placements is computationally possible. Additionally, since this initial formulation is very computationally efficient, it opens the door to many extensions designed to improve the sensor placements while still being solvable in a tractable amount of time.

This optimal placement formulation relies on the data provided by CFD dispersion tools. This scenario generation can be extremely computationally expensive, requiring several hours and even days to generate just one scenario. Because these scenarios are so expensive, it may be beneficial to design extensions to the problem formulation to make best use of the limited scenario data available. Furthermore, we would like a method by which to determine how much confidence can be placed in the data set used for the optimization, as well as a way to determine when enough scenarios have been utilized.

While this problem formulation provides optimal values for the mean detection time across all scenarios, it provides no guarantee of tail behavior through the dis-

tribution of detection times for all scenarios. More clearly, this means that while the mean detection time for all scenarios may be very low, a few of the scenarios may require much longer time periods to detect. This longer time period could be of much more concern to the designers as safety systems because these scenarios may have much worse consequences. A min-max formulation has been solved to minimize the worst-case scenario, but this provides not guarantee that the rest of the distribution will even be close to optimal. An extension will be presented that considers conditional-value-at-risk to determine the optimal placement for the entire distribution while also providing control over the tail behavior of the distribution.

These results show that the use of MILPs for the placement of gas detectors is possible, and that with the proper extensions, can be an effective tool for aiding the design of safety systems throughout the petrochemical industry. This technique provides a systematic method for determining optimal gas detector placements while making full use of available gas dispersion scenarios, representing a significant step forward in the methods currently used in the petrochemical industry. Further extensions to this formulation will be examined in future chapters.

3. COVERAGE CONSTRAINTS FOR IMPROVED DETECTOR PLACEMENTS*

If the number of scenarios used for the sensor placement is not significantly larger than the number of sensors to be placed or the scenario space has not sufficiently been sampled, the optimal placement using the basic formulation presented above can be biased towards the specific scenarios considered and may not be resilient in the face of alternate scenarios not considered in the optimization. Therefore, a second formulation is also developed that includes an additional constraint to guarantee a particular volumetric coverage and improve detectability in areas of the facility that are underrepresented by the given set of scenarios. This constraint hybridizes traditional heuristic-based coverage approaches with our optimization based approach. The second mixed-integer formulation, labeled problem (SPC), includes the same set of equations from Problem (SP), but contains additional constraints on the maximum allowable distances between any two placed gas detectors.

The proposed problem formulation is presented in the following section. Numerical results for the optimal placement of gas detectors by Problem (SP), Problem (SPC), and a coverage-only placement (C) is presented in Section 3.2. In this chapter, the full scenario set is considered when generating the proposed sensor placements; however, in Chapter 5, a sampling procedure is utilized to simulate unforeseen scenarios when evaluating these placements. This sampling procedure demonstrates that the formulation considering coverage is in fact more resilient to unforeseen scenarios. In Section 3.3, a summary of this chapter, as well as some conclusions and

*Part of this section is reprinted with permission from "A Stochastic Programming Approach For Gas Detector Placement Using CFD-based Dispersion Simulations" by Legg SW, Benavides-Serrano AJ, Siirola JD, Watson JP, Davis SG, Bratteteig A, and Laird CD, 2012. Computers & Chemical Engineering, 47(0):194-201, Copyright 2012 by Elsevier Ltd.

discussion, is presented.

3.1 Formulation Including Coverage Constraints

Problem (SPC) includes equations (3.1a-3.1f) from Problem (SP) along with the additional constraints on coverage (3.1g),

$$\min \quad \sum_{a \in A} \alpha_a \sum_{i \in \mathcal{L}_a} d_{a,i} x_{a,i} \tag{3.1a}$$

s.t.

$$\sum_{l \in L} s_l \leq p \tag{3.1b}$$

$$x_{a,i} \leq s_i \qquad \forall\, a \in A,\, i \in \mathcal{L}_a \tag{3.1c}$$

$$\sum_{i \in \mathcal{L}_a} x_{a,i} = 1 \qquad \forall\, a \in A \tag{3.1d}$$

$$s_l \in \{0, 1\} \qquad \forall\, l \in L \tag{3.1e}$$

$$0 \leq x_{a,i} \leq 1 \qquad \forall\, a \in A,\, i \in \mathcal{L}_a \tag{3.1f}$$

$$\sum_{c \in C_l} s_c \geq 1 \qquad \forall\, l \in L, \tag{3.1g}$$

where C_l is the set of all locations that provide coverage to location l. This constraint requires every location in the facility to be provided coverage by at least one placed sensor. Coverage sets are created in a preprocessing step. A coverage distance is first specified. Then, for each location l, the set C_l is defined as all of the potential detector locations that are within the coverage distance. While the use of sensor locations seems unrelated to the definition of a coverage area, in our examples, the candidate sensor locations include an equally-spaced, three-dimensional grid across the entire facility. The intent of this hybridized approach is to meet the coverage goals with the additional benefit of intelligent placement of sensors based on rigorous

simulation data from modern dispersion simulation tools.

3.2 Numerical Results

3.2.1 Data Set 1 Results

To perform a comparison of the solutions resulting from different placement formulations, we set p=50 and solve problems (SP) and (SPC) using all available scenarios. For problem (SPC), we enforce a coverage distance of 6 m. To produce a fair comparison against a pure coverage-based approach, we find a placement that provides the best coverage possible given 50 sensors. This placement is labeled (C) and, for this case study, a coverage distance of 5.5 m was possible. Note that this placement may not be unique (there may be other placements of 50 sensors that also provide a 5.5 m coverage distance), however this placement achieves a coverage that is better than that required by problem (SPC).

	(SP)	(SPC)	(C)
Exp. Detection Time (s)	26.41	31.38	149.9
Fraction Detected	1.000	1.000	0.800

Table 3.1: Data Set 1: Results for 3 formulations using 50 sensors

Table 3.1 shows the performance of the gas detector placements found using each of these three approaches. Problem (SPC) has an additional coverage constraint and, as expected, the optimal expected detection time is worse than that using the placement from problem (SP). However, given the coverage constraint, we expect formulation (SPC) to be more resilient to leak scenarios not considered in the optimization. Note also that the optimal expected detection time is less than 5 seconds

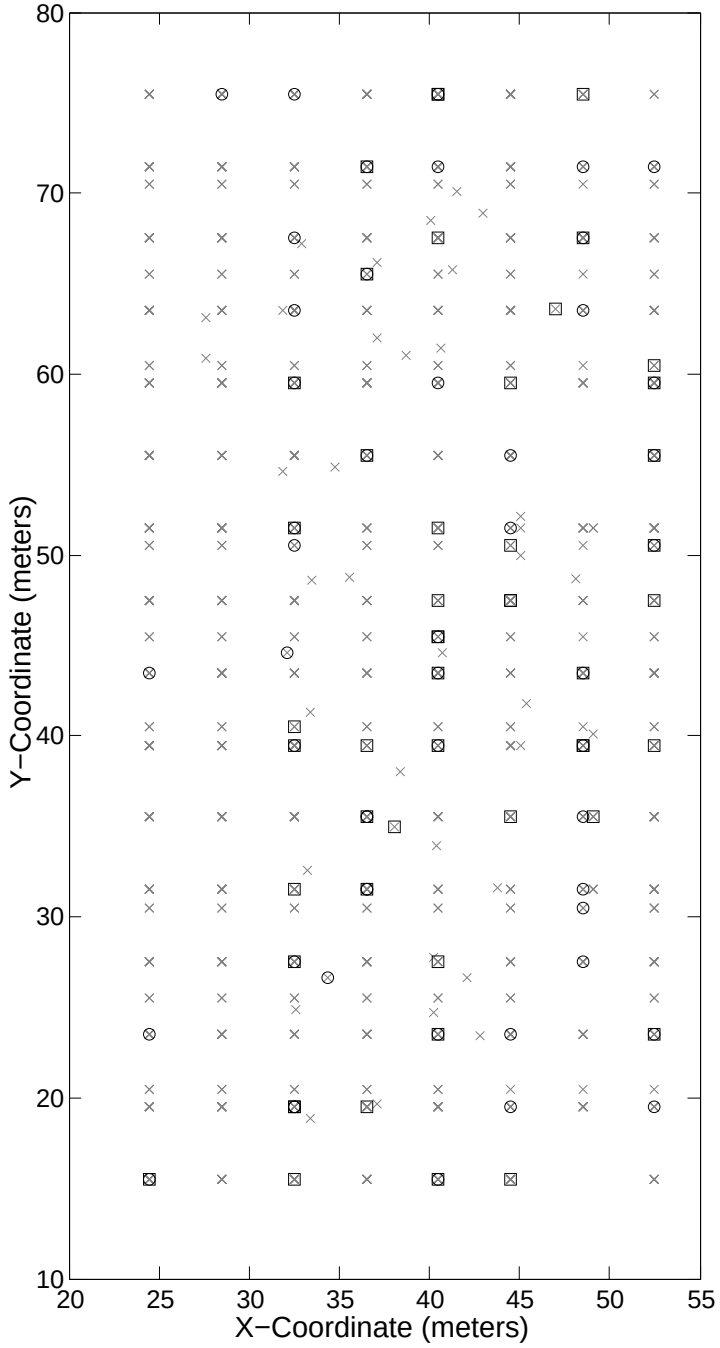

Figure 3.1: Locations of sensors placed by (SP) and (SP-C) in process facility for Data Set 1. $\times$ represent potential sensor locations, $\square$ represent sensors placed by formulation (SP), and $\circ$ represent sensors placed by formulation (SP-C).

longer, and both formulations are able to detect all the leak scenarios considered. This is in stark contrast to the coverage-only placement (C), where the expected detection time was over 5 times greater, and only four in five leak scenarios are detected. Even though the coverage provided by (C) is tighter than that required by problem (SPC), the performance of the coverage-only approach is significantly worse on these leak scenarios.

In Figure 3.1, we can see how the placement of sensors has been affected by the addition of a coverage constraint. In this figure, the sensors locations chosen by Problem (SP) (as shown in Chapter 2) are shown by the $\square$ markings, while the new locations, as chosen by Problem (SPC), are shown by the $\circ$ markings. While Problem (SP) focused solely on locating sensors based on the set of scenarios used for the optimization problem, the coverage constraints in Problem (SPC) generate a more cautious coverage of the entire facility space by locating sensors more evenly across the set of potential locations. As we can see, the placements generated by these two formulations vary in approximately 40% of the locations chosen.

Additional results for Problem (SPC) are shown in Chapter 5. Results are shown for the performance of Problem (SP), Problem (SPC), and the coverage-only placement (C) on unforeseen scenarios through the use of a sampling procedure. Optimal placements are determined using a subset of scenarios from the entire scenario set, and additional subsets of the full scenario set are used to evaluate the placement.

3.2.2 Data Set 2 Results

A comparison of the solutions resulting from different placement formulations by allocating p=30 sensors for each formulation and solve problems (SP) and (SPC) using all available scenarios. For data set 2, we enforce a coverage distance of 4 m for Problem (SPC). To produce a fair comparison against a pure coverage-based

approach, we find a placement that provides the best coverage possible given 30 sensors. This placement is labeled (C) and, for this case study, a coverage distance of 3.5 m was possible. Note that this placement may not be unique (there may be other placements of 30 sensors that also provide a 3.5 m coverage distance), however this placement achieves a coverage that is better than that required by problem (SPC).

	(SP)	(SPC)	(C)
Exp. Detection Time (s)	17.29	18.82	44.64
Fraction Detected	1.000	1.000	0.981

Table 3.2: Data Set 2: Results for 3 formulations using 30 sensors

Table 3.2 shows the performance of the gas detector placements found using each of these three approaches. Problem (SPC) has an additional coverage constraint and, as expected, the optimal expected detection time is worse than that using the placement from problem (SP). However, given the coverage constraint, we expect formulation (SPC) to be more resilient to alternate leak scenarios. Note also that the optimal expected detection time is less than 1.5 seconds longer, and both formulations are able to detect all the leak scenarios considered. The coverage-only placement (C) resulted in an expected detection time was over 2.5 times greater. However, unlike the performance of (C) on data set 1 where only 80% of scenarios were detected, over 98% of events were detected for data set 2. The area of the facility was much smaller, and the tighter coverage distance allowed this result to be much better. The placements generated by (SP) and (SPC) detected all of the release events. The

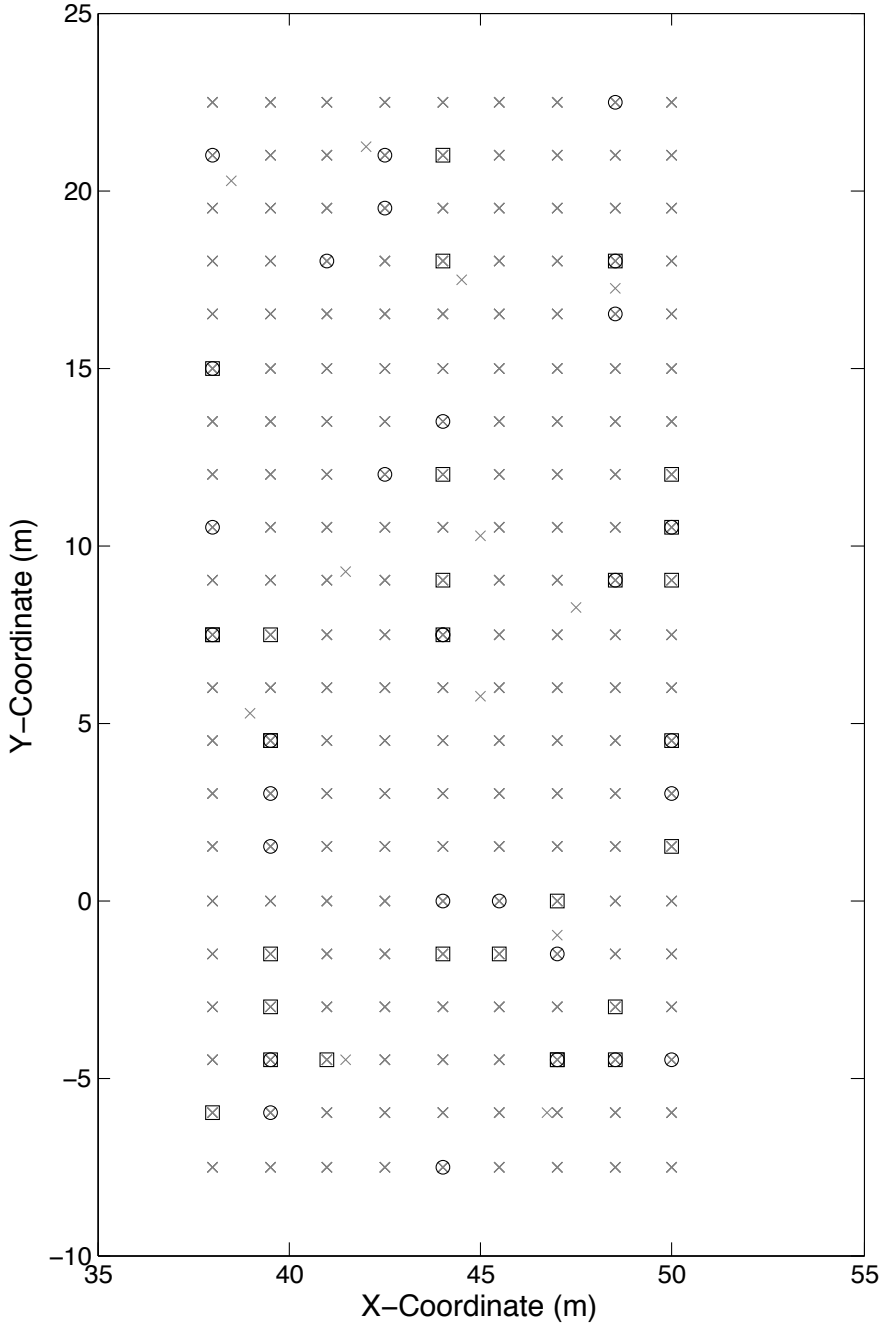

Figure 3.2: Locations of sensors placed by (SP) and (SP-C) in process facility for Data Set 2. × represent potential sensor locations, □ represent sensors placed by formulation (SP), and ∘ represent sensors placed by formulation (SP-C).

optimal placements are still better than the coverage-only placement (C), even given then much smaller area.

Figure 3.2 shows how the placement of sensors has been altered by the inclusion of the coverage constraint. The sensor locations chosen by Problem (SP) (as shown in Chapter 2) are shown by the $\square$ markings, while the locations selected by Problem (SPC) are shown by the $\circ$ markings. Problem (SPC) focuses on optimally placing sensor while also being more cautious and allocating sensors more evenly throughout the facility. Approximately 50% of the sensors are different between the two placements. As we can see, the inclusion of the coverage constraint has a significant impact on the placement of sensors.

3.2.3 Data Set 3 Results

For data set 3 we compare the solutions from each of the sensor placement formulations by limiting the number of sensors place to p=55 and solve problems (SP) and (SPC) using all available scenarios. For problem (SPC), we enforce a coverage distance of 5 m. To produce a fair comparison against a pure coverage-based approach, we find a placement that provides the best coverage possible. This placement is labeled (C) and, for this case study, a coverage distance of 4.5 m was possible. Note that this placement may not be unique (there may be other placements that provide a 4.5 m coverage distance), however this placement achieves a coverage that is better than that required by problem (SPC). The placement generated by (C) includes two more sensors than either (SP) or (SPC). Therefore, it should have a slight advantage over those two placements.

Table 3.3 shows the performance of the gas detector placements found using each of these three approaches. Problem (SPC) has an additional coverage constraint and, as expected, the optimal expected detection time is worse than that using the

	(SP)	(SPC)	(C)
Exp. Detection Time (s)	17.99	19.36	90.80
Fraction Detected	1.000	1.000	0.883

Table 3.3: Data Set 3: Results for 3 formulations using 55 sensors

placement from problem (SP). However, given the coverage constraint, we expect formulation (SPC) to be more resilient to alternate leak scenarios. Note also that the optimal expected detection time is less than 2 seconds larger, and both formulations are able to detect all the leak scenarios considered. The coverage-only placement (C) resulted in an expected detection time was over 4 times greater than either Problem (SP) or (SPC). The placement generated by (C) also only detects roughly 88% of the scenarios, performing much worse than the two optimal placements. These results are obtained even though (C) has a tighter coverage than either of the optimization based placements. This shows how the use of optimization techniques can improve sensor placements.

Figure 3.3 shows the two sensor placements generated by Problem (SP) and Problem (SPC) for data set 3. The sensor locations chosen by Problem (SP) (as shown in Chapter 2) are shown by the $\square$ markings, while the locations selected by Problem (SPC) are shown by the $\circ$ markings. Problem (SPC) focuses on optimally placing sensor while also being more cautious and allocating sensors more evenly throughout the facility. Approximately 40% of the sensors are different between the two placements. As we can see, the inclusion of the coverage constraint has a significant impact on the placement of sensors.

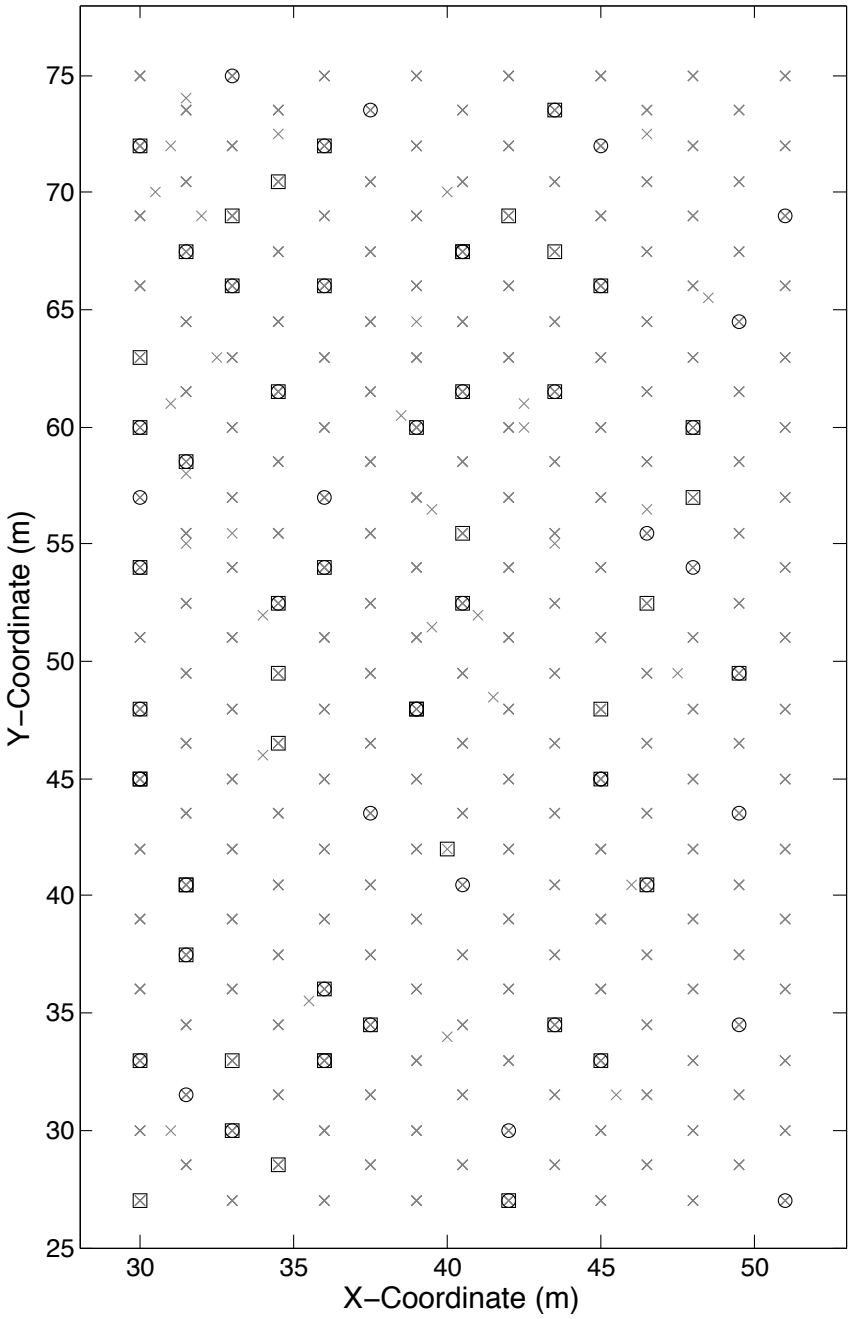

Figure 3.3: Locations of sensors placed by (SP) and (SP-C) in process facility for Data Set 3. $\times$ represent potential sensor locations, $\square$ represent sensors placed by formulation (SP), and $\circ$ represent sensors placed by formulation (SP-C).

3.3 Summary

In this chapter, an extension to the original formulation, Problem (SP), was proposed. This extension involved the addition of a constraint on the maximum distance between potential sensor locations and the nearest selected sensor location. This constraint enforces a coverage throughout all areas of the facility, regardless of whether or not the data set has scenarios impacting those areas. This combats any problems that may arise by a scenario set that does not fully represent the entire uncertainty range for potential release locations and dispersions. Additionally, this constraint is a hybridization of current industry techniques for provide some level of coverage throughout a facility. This formulation represents the combination of industry techniques with optimization methods.

Numerical results were provided for three independent data sets. These results show that Problem (SPC) suffers from a minimal increase in expected detection time across the scenarios as compared to Problem (SP). Additionally, we see that it detects all of the scenarios, like the optimal placement provided by Problem (SP). An overhead view of the facility shows how the location of sensors by these two placement formulations actually differs, but yields similar optimal values. This formulation provides a provably optimal solution while also being computationally efficient. For the size of problem considered, the results required a few seconds on a dual quad-core Intel(R) Xeon(R) CPU X5482 with a clock speed of 3.2GHz and 18 GB RAM. Solution time is similar on a standards dual-core laptop or desktop. These results show that we can add a coverage constraint without dramatically impacting the optimal objective value. In Chapter 5, we perform optimizations over different sets of subsamples and show that the coverage formulation is indeed more resilient to unforeseen scenarios.

4. INCORPORATING CONDITIONAL-VALUE-AT-RISK FOR IMPROVED TAIL BEHAVIOR*

Our previous work has focused on minimizing the expected detection time (Legg et al., 2012a,b). However, the optimal sensor placement for this objective can yield poor worst-case performance. This is illustrated by the probability density functions shown in Figure 4.1. The dashed curve may have acceptable mean behavior, but excessively large detection times in the tail. While the probability of realizing a detection time in this tail is low, the consequences may be unacceptable. This is in contrast to the probability distribution shown by the solid line. Here, the mean detection time is larger, but the worst-case and tail-behavior are significantly improved. These observations motivate the application of Conditional Value-at-Risk (CVaR) as an alternate risk metric in our formulation. CVaR is a popular downside risk measure with desirable theoretical and computational properties. By constraining CVaR, the optimization formulation produces detector placements with significantly improved tail-behavior. This work can by found in the paper by Legg et al. (2013).

In order to improve the tail behavior, we extend our previous work and add an additional constraint on Conditional-Value-at-Risk. CVaR is defined based on another popular risk measure Value-at-Risk (VaR). VaR has been widely applied and studied in the 1990s (Duffie and Pan, 1997). Figure 4.2 shows a visual interpretation of VaR. In the context of this paper, VaR indicates the largest detection time we might expect within a specified level of confidence. Specifically, for a random variable

*Part of this section is reprinted with permission from "Optimal Gas Detector Placement Under Uncertainty Considering Conditional-Value-at-Risk" by Legg SW, Wang C, Benavides-Serrano AJ, and Laird CD, 2013. Journal of Loss Prevention in the Process Industries, 26(3):410-417, Copyright 2013 by Elsevier Ltd.

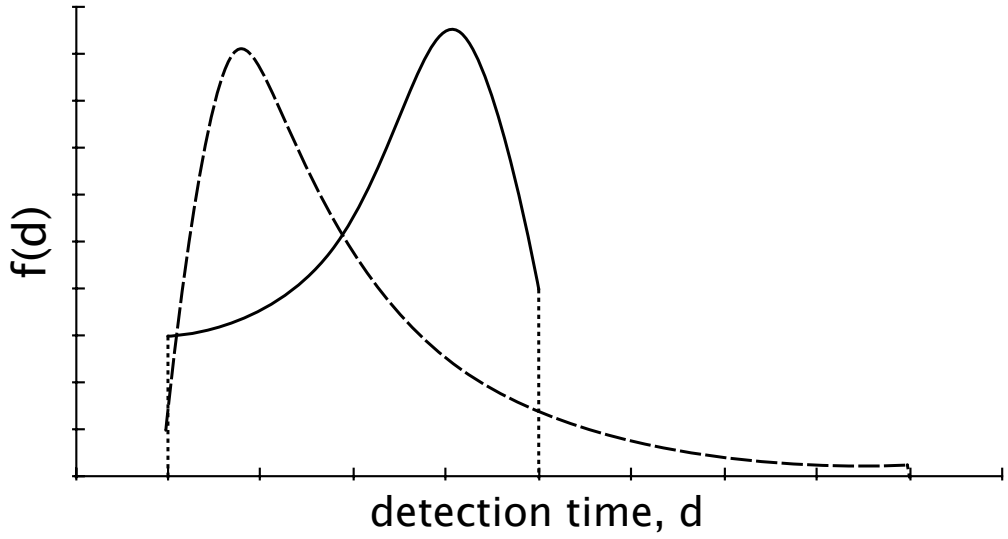

Figure 4.1: Probability density functions for optimal detection times with different mean and tail behavior

$\tilde{t}$ and $0 < \theta < 1$,

$$VaR_\theta := \min\{\varsigma \in \mathbb{R} : \int_{\tilde{t} \leq \varsigma} f(\tilde{t})\, \mathrm{d}\tilde{t} \geq \theta\} \tag{4.1}$$

where $f(\tilde{t})$ is the density function of $\tilde{t}$. Although VaR is a popular risk measure, it is not preferred since it lacks desirable mathematical characteristics such as subadditivity and convexity, and hence is less appealing for the control and optimization of risk (Krokhmal et al., 2011). A measure based on VaR, known as CVaR, was proposed in the early 2000s (Rockafellar and Uryasev, 2000; Uryasev, 2000; Rockafellar and Uryasev, 2002). CVaR, as shown in Figure 4.2, is the mean value of the detection time, conditional on the detection time exceeding VaR. In this sense, VaR provides the threshold value of the tail, while CVaR provides the mean value of the

tail. CVaR is defined as

$$CVaR_\theta := (1-\theta)^{-1} \int_{\tilde{t} \geq VaR_\theta} \tilde{t}\ f(\tilde{t})\ \mathrm{d}\tilde{t}, \tag{4.2}$$

and exhibits numerous desirable mathematical properties (Artzner et al., 1999). Furthermore, CVaR is not as difficult to compute as it might appear given definition (4.2) since it has been proven that

$$CVaR_\theta = \min_\beta F_\theta(\beta) \tag{4.3}$$

where

$$F_\theta(\beta) := \beta + (1-\theta)^{-1} \int [\tilde{t} - \beta]^+ f(\tilde{t}) \mathrm{d}\tilde{t} \tag{4.4a}$$

$$[\tilde{t} - \beta]^+ = \max\{0, \tilde{t} - \beta\}. \tag{4.4b}$$

4.1 CVaR Problem Formulation

By (4.4a), the CVaR of detection time $\tilde{t}$, given sensor placement s, is

$$\begin{aligned} &CVaR_\theta(\tilde{t}(s)) = \\ &\min_\beta \quad \beta + (1-\theta)^{-1} \sum_{a \in A} \alpha_a [t(a,s) - \beta]^+ \end{aligned} \tag{4.5}$$

where $t(a,s)$ is the detection time for scenario a given by $\sum_{i \in \mathcal{L}_a} d_{a,i} x_{a,i}$, and θ is the desired confidence level. An optimization formulation for minimizing the CVaR of

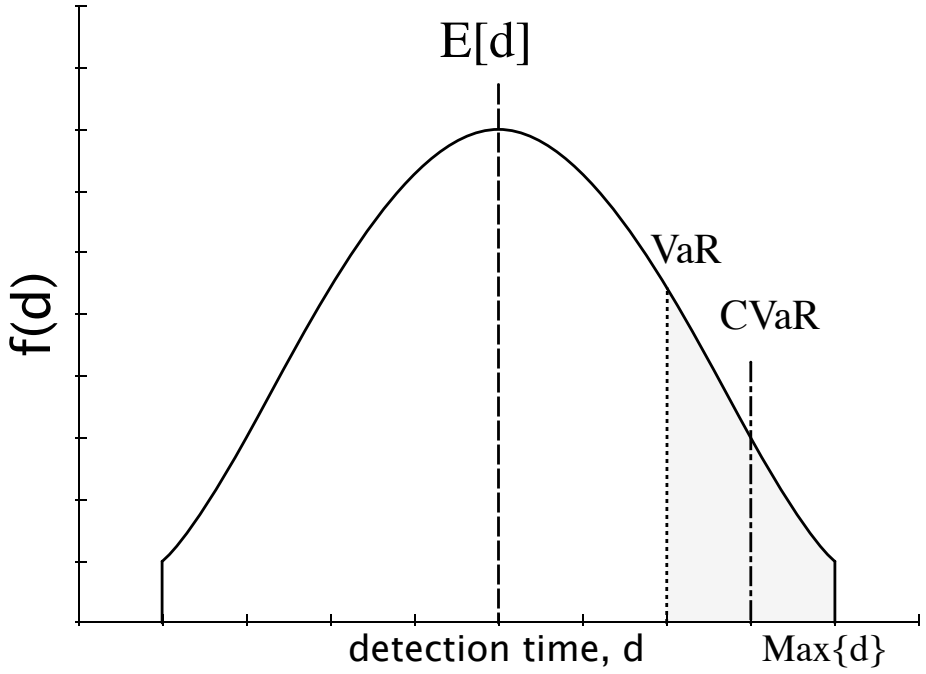

Figure 4.2: Visualization of VaR and CVaR

detection time, referred to in this paper as problem (CVaR), can be written as

$$\min \quad \beta + \frac{1}{1-\theta} \sum_{a \in A} z_a \alpha_a \tag{4.6a}$$

s.t.

$$z_a \geq 0 \qquad \forall a \in A \tag{4.6b}$$

$$z_a \geq \sum_{i \in \mathcal{L}_a} d_{a,i} x_{a,i} - \beta \qquad \forall a \in A \tag{4.6c}$$

$$\sum_{l \in L} s_l \leq p \tag{4.6d}$$

$$x_{a,i} \leq s_i \qquad \forall\, a \in A,\, i \in \mathcal{L}_a \tag{4.6e}$$

$$\sum_{i \in \mathcal{L}_a} x_{a,i} = 1 \qquad \forall\, a \in A \tag{4.6f}$$

$$s_l \in \{0, 1\} \qquad \forall\, l \in L \tag{4.6g}$$

$$0 \leq x_{a,i} \leq 1 \qquad \forall\, a \in A,\, i \in \mathcal{L}_a. \tag{4.6h}$$

While this formulation solves to an optimal placement that has improved tail behavior, as expected, there can be significant non-uniqueness in the solution (i.e. several different sensor layouts give the same optimal CVaR value). Therefore, we use this formulation to determine a target value for CVaR, then exploit the nonuniqueness to minimize expected detection time subject to this constraint on CVaR. This can be achieved by adding a constraint on the maximum allowable CVaR value to problem formulation (SP). This new formulation, called problem (SP-CVaR) in this paper, requires the addition of constraints (4.7g-4.7i) to problem (SP) (4.7a-4.7f):

$$\min \quad \sum_{a\in A} \alpha_a \sum_{i\in\mathcal{L}_a} d_{a,i}x_{a,i} \tag{4.7a}$$

s.t.

$$\sum_{l\in L} s_l \leq p \tag{4.7b}$$

$$x_{a,i} \leq s_i \qquad \forall\, a \in A,\, i \in \mathcal{L}_a \tag{4.7c}$$

$$\sum_{i\in\mathcal{L}_a} x_{a,i} = 1 \qquad \forall\, a \in A \tag{4.7d}$$

$$s_l \in \{0,1\} \qquad \forall\, l \in L \tag{4.7e}$$

$$0 \leq x_{a,i} \leq 1 \qquad \forall\, a \in A,\, i \in \mathcal{L}_a \tag{4.7f}$$

$$\beta + \frac{1}{1-\theta}\sum_{a\in A} z_a\alpha_a \leq CVaR^* \tag{4.7g}$$

$$z_a \geq 0 \qquad \forall a \in A \tag{4.7h}$$

$$z_a \geq \sum_{i\in\mathcal{L}_a} d_{a,i}x_{a,i} - \beta \qquad \forall a \in A. \tag{4.7i}$$

The target value $CVaR^*$ is determined by a prior solution of problem (CVaR).

4.2 Numerical Results

Given this data, problem (SP) represents a stochastic programming problem that solves for an optimal placement of gas detectors that minimizes the expected value of the detection time across the scenarios considered. Problem (CVaR) represents the stochastic programming problem designed to optimally place gas detectors in order to minimize CVaR of the set of scenarios. This formulation provides a target value for the upper bound constraint on CVaR for the third formulation, problem (SP-CVaR). Problem (SP-CVaR) chooses an optimal gas detector placement scheme that minimizes the expected detection time for all scenarios while constraining CVaR to be less than or equal to the target value. Results will be shown below for the three data sets.

4.2.1 Data Set 1 Results

Table 4.1 presents the results for the solution of the stochastic programming formulations (SP) and (SP-CVaR). Each of these problems was solved, allowing a maximum of p=50 sensors to be placed throughout the facility. Note that in each case, all of the 270 scenarios were detected. Given the optimal placement from each formulation, we calculated the detection time across all scenarios. This table shows the minimum, mean, and maximum detection times. As expected, problem (SP) provides the best expected detection time of approximately 33 seconds. However, the worst-case behavior is significantly higher than the expected value at approximately 110 seconds. This gap motivates the use of programming formulations incorporating measures of tail behavior. Problem (SP-CVaR), was solved for five different values of θ. A target value for the CVaR upper bound was determined by solving problem formulation (CVaR) presented in Section 4.1. At large values of θ, the solution to (SP-CVaR) produces significantly improved tail behavior, lowering the worst-case

detection time to 81 seconds, nearly 30 seconds better than that produced by problem (SP). Furthermore, in this example formulation (SP-CVaR) sacrifices little in terms of the expected detection time. As the confidence level θ is reduced, formulation (SP-CVaR) produces results that resemble those of problem (SP). At these lower values of θ, the distribution that makes up VaR includes almost all of the probability distribution function, and in the limit as $\theta \to 0$ it approaches the minimum expected detection time formulation.

	Min	Mean	Max
Problem (SP)	20.03	33.18	110.5
Problem (SP-CVaR)			
θ=0.95	20.03	34.55	81.01
θ=0.90	20.03	34.55	81.01
θ=0.80	20.03	34.08	81.01
θ=0.20	20.03	33.18	110.5
θ=0.10	20.03	33.18	110.5
Coverage Only	22.00	56.61	266.7

Table 4.1: Data Set 1: Minimum, maximum, and mean values for the damage coefficients of each problem formulation.

Figure 4.3 displays a histogram of the detection times for the optimal sensor placements. In the top figure, the detection times for problem (SP) are presented. As expected, this formulation produces a placement that includes a bulk of the scenarios towards the low end of the distribution, as indicated by the high number

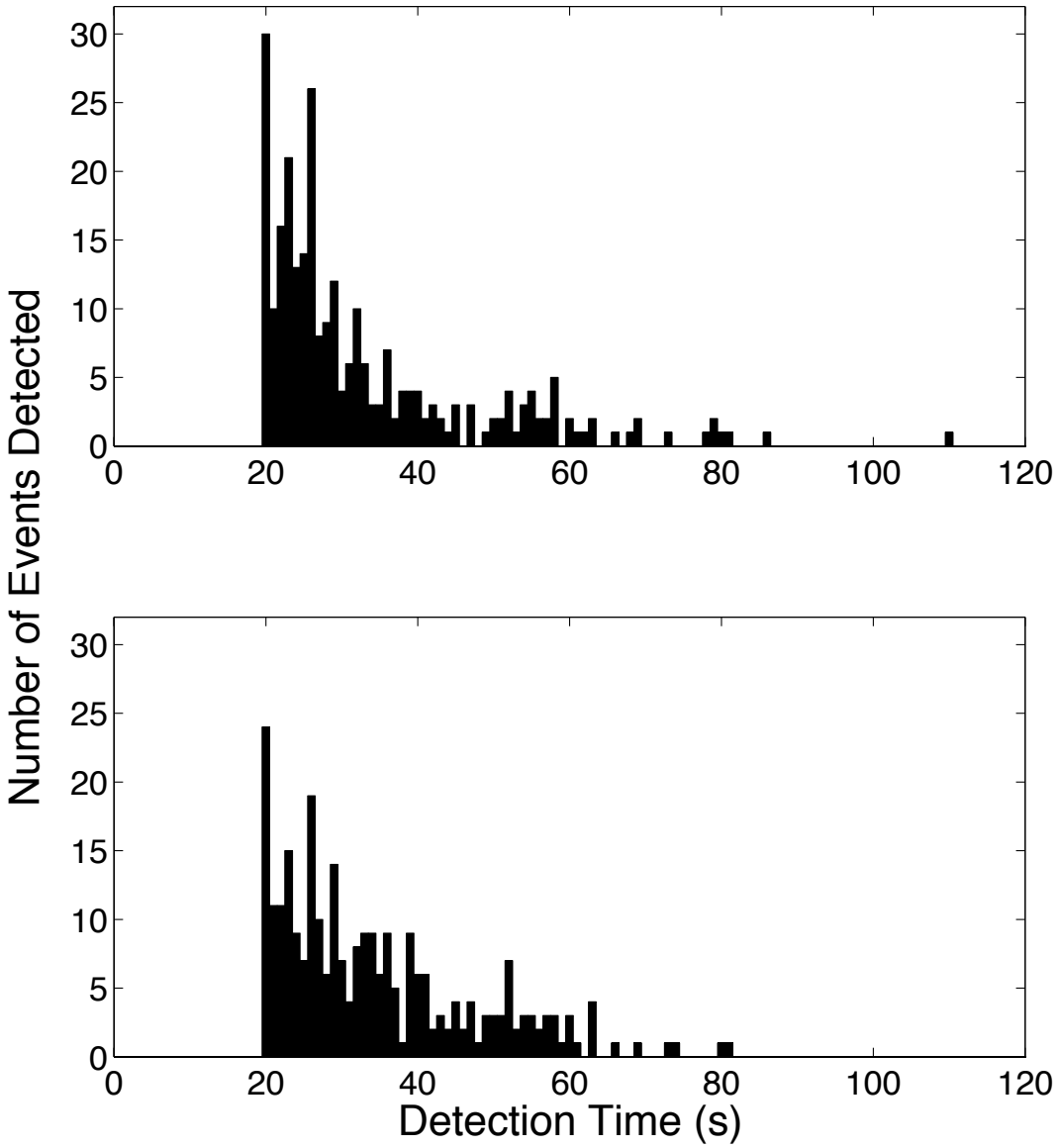

Figure 4.3: Expected detection times for (SP) and (SP-CVaR) for Data Set 1. Expected detection times for all of the scenarios in the solution of problem SP (top) and problem SP-CVaR (bottom) where θ=0.95.

of events located in the 20 second to 40 second range. While the mean detection time is minimized, it is also apparent that this problem formulation does not explicitly consider the tail of the distribution, sacrificing lower detection times for a few events in order to improve the mean. Several events were detected near 80 seconds, and the worst-case detection time was 110.5 seconds. However, problem (SP-CVaR) produces an optimal placement that corrects this behavior, as shown in the bottom panel. Problem (SP-CVaR) was formulated with a limit on the CVaR for the 95%-level tail distribution ($\theta = 0.95$). The worst-case scenario has been reduced to approximately 80 seconds, as also indicated in Table 4.1. Only 6 scenarios required more than 65 seconds to detect, which is half of the number of events in that range from the placement produced by (SP). This small sacrifice in expected detection time is acceptable in instances where it is desired to reduce the number of high consequence scenarios. To provide a representative comparison, Table 4.1 also shows results for a placement that was based on coverage alone. We assume that each sensor provides a fixed coverage radius r. Then, we determine the tightest coverage radius possible while ensuring that each monitoring point is protected by at least one sensor. Limiting the number of sensors to be no more than p=50, in essence, this produces a placement that maximizes the coverage that is possible using the 50 sensors. In Table 4.1 we see that this coverage-only approach performs significantly worse with a mean detection time of approximately 56 seconds, and a worst-case detection time of approximately 266 seconds, over 4 times worse than the CVaR formulation. Furthermore, this placement produces very high detection times and a full 45% of all simulated leak scenarios were not detected. (Note that the min, mean, and worst-case numbers given for the coverage-only case included the detected scenarios only.)

Finally, Figure 4.4 presents an overhead view of the sensor placement layouts

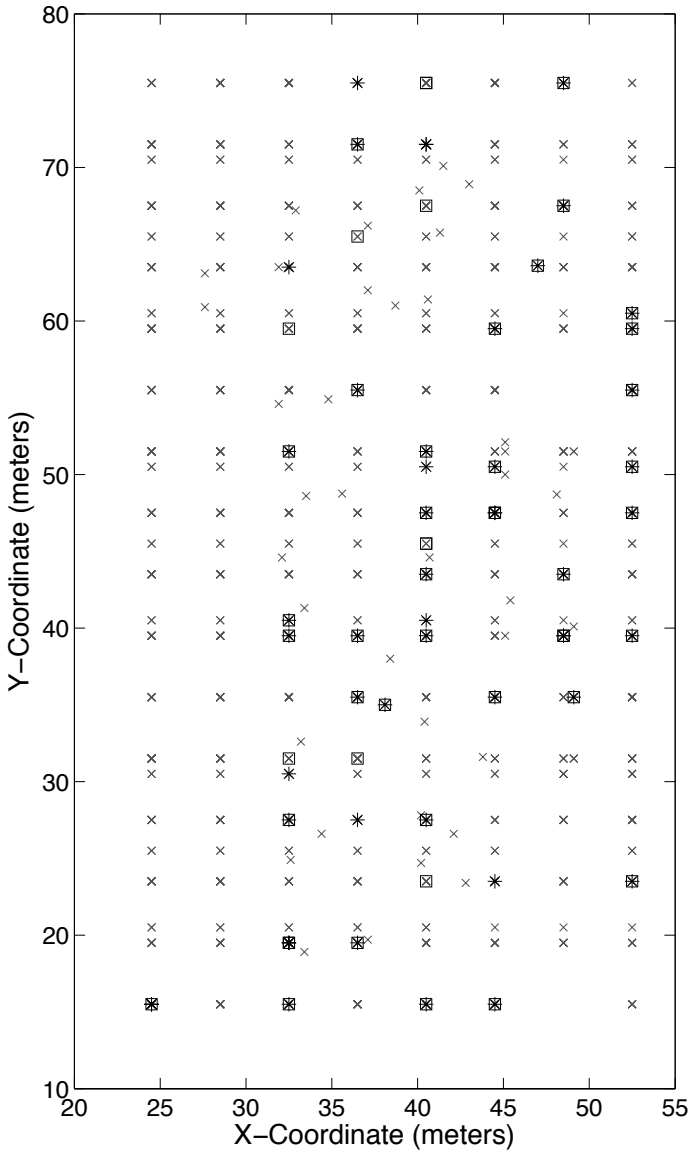

Figure 4.4: Locations of sensors placed by (SP) and (SP-CVaR) in process facility for Data Set 1. $\times$ represent potential sensor locations, $\square$ represent sensors placed by formulation (SP), and $*$ represent sensors placed by formulation (SP-CVaR). The value of θ=0.95.

produced by problem (SP) and problem (SP-CVaR). Both the x and y facility coordinates are provided in meters, and all of the vertical z coordinates have been collapsed into one layer. The $\times$ symbols represent all of the potential sensor locations. The sensors placed by problem (SP) are represented by the $\square$ symbols, while those placed by problem (SP-CVaR) are marked with the $*$ symbol. Of the 50 sensors placed by each formulation, 42 were placed in the same location. This means that 8 sensors were moved based only upon the addition of the CVaR upper bound constraint. Therefore, while the expected value of the detection time changes little, there are significant differences in the sensor placement found.

4.2.2 Data Set 2 Results

Table 4.2 presents the results for the solution of the stochastic programming formulations (SP) and (SP-CVaR) using data set 2. Each of these problems is solved with p=30 sensors to be placed throughout the facility. Note that in each case, all of the 314 scenarios in data set 2 were detected. Considering detection time across all scenarios, this table shows the minimum, mean, and maximum detection times. Problem (SP) provides the best expected detection time at 17.29 seconds. The worst detection time is significantly higher at 518.5 seconds. Problem (SP-CVaR) was solved using five different values of θ (as seen in Table 4.2). The target CVaR value for the constraint in Problem (SP-CVaR) was determined by solving Problem (CVaR), as shown in Section 4.1. Larger values of θ should provide much improved tail behaviour, while lower values should result in solutions that are much more similar to Problem (SP). As we can see, the Problem (SP-CVaR) sacrifices very little in terms of the expected detection time for all values of θ. However, it appears that the longest detection time required for any scenario does not change by considering CVaR in the problem formulation. This demands a closer look at the distribution of

the detection times.

	Min	Mean	Max
Problem (SP)	15.05	17.29	518.5
Problem (SP-CVaR)			
θ=0.95	15.05	17.32	518.5
θ=0.90	15.05	17.30	518.5
θ=0.80	15.05	17.30	518.5
θ=0.20	15.05	17.29	518.5
θ=0.10	15.05	17.29	518.5

Table 4.2: Data Set 2: Minimum, maximum, and mean values for the damage coefficients of each problem formulation.

In Figure 4.5, the distribution of the detection times for each scenario is shown for each of the optimal sensor placements. In the top panel, the detection times for Problem (SP) are shown. This formulation produces a placement that has a bulk of the scenarios detected in a time towards the low end of the distribution. A majority of the scenarios are detected between 15 and 17 seconds. This problem formulation does an effective job minimizing the expected detection time across all of the scenarios, but it is also apparent that there are several scenarios that require more than 18 seconds to detect. Problem (SP-CVaR) explicitly considers the events in the tail of the distribution when determining the optimal placement of sensors. When Problem (SP-CVaR) is formulated with a limit on the CVaR for the 95%-level tail distribution, *i.e.* θ=0.95, roughly 35% fewer scenarios are detected above the 18 second mark. There is very little sacrifice in the mean detection time to improve

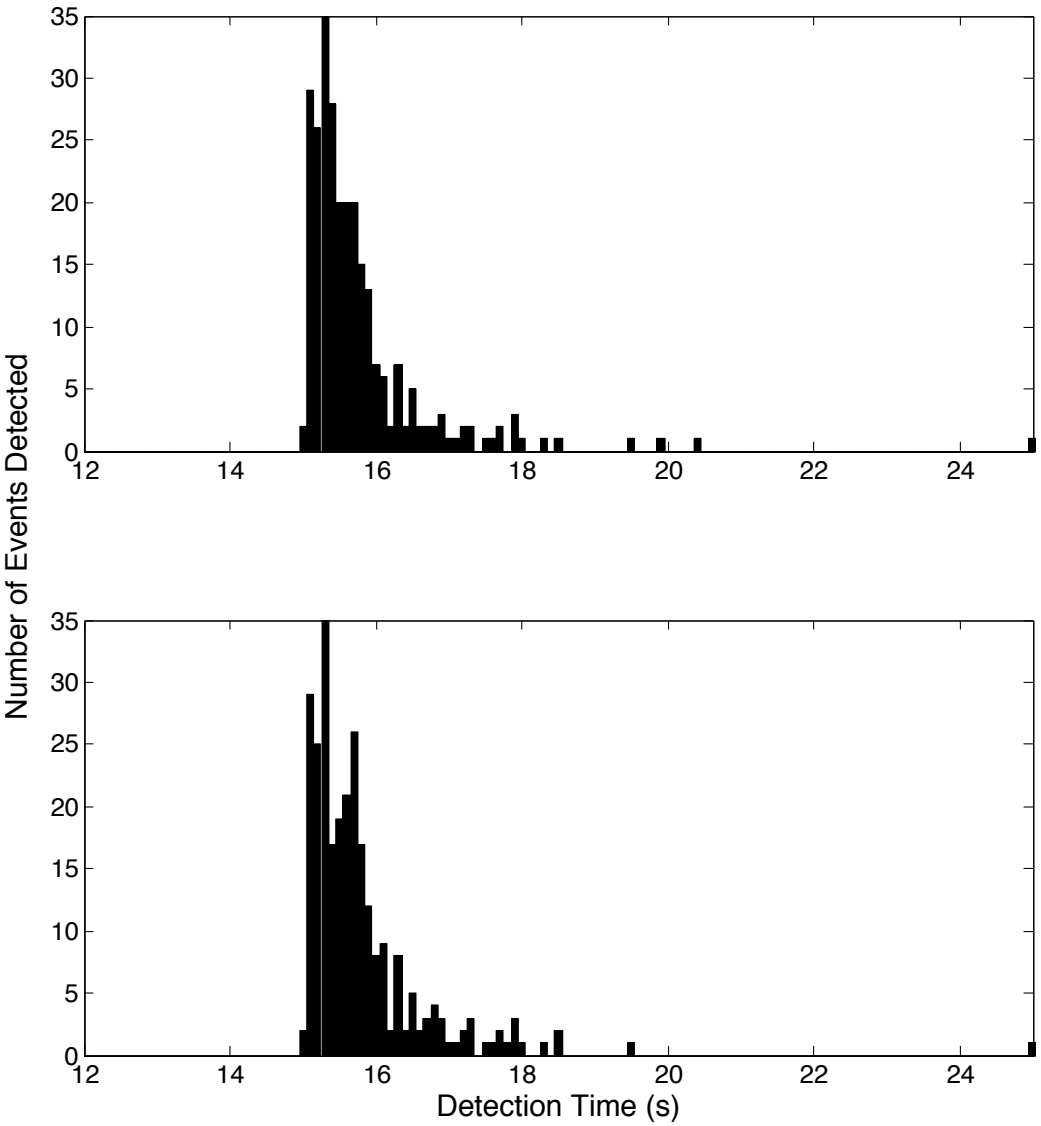

Figure 4.5: Expected detection times for (SP) and (SP-CVaR) for Data Set 2. Expected detection times for all of the scenarios in the solution of problem SP (top) and problem SP-CVaR (bottom) where θ=0.95.

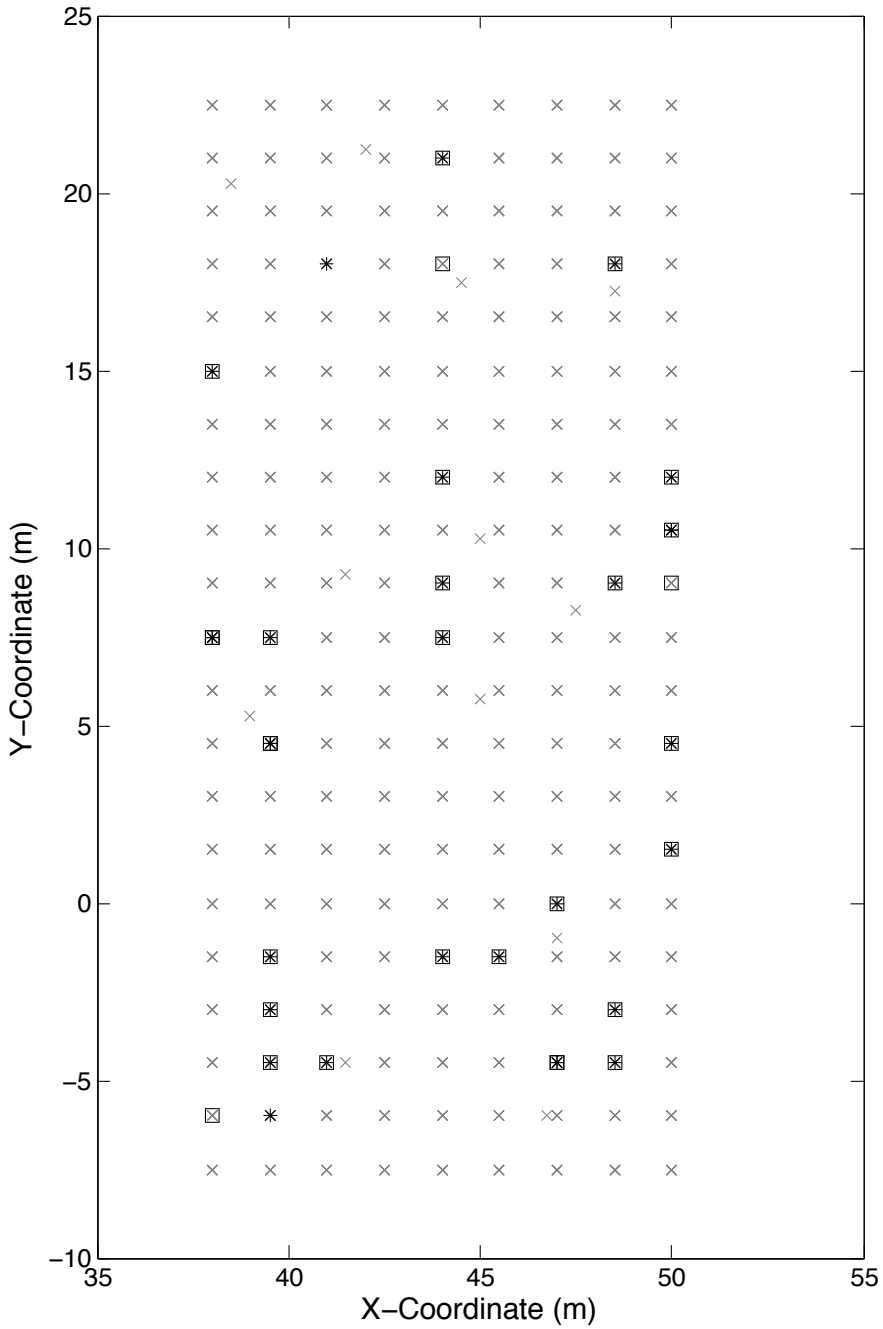

Figure 4.6: Locations of sensors placed by (SP) and (SP-CVaR) in process facility for Data Set 2. $\times$ represent potential sensor locations, $\square$ represent sensors placed by formulation (SP), and $*$ represent sensors placed by formulation (SP-CVaR). The value of θ=0.95.

this tail behavior. This can be inspected visually by noting how the bulk of the distribution shifts slightly to the right as compared to the distribution of Problem (SP). In both figures, there is one scenario that is detected in 518.5 seconds. This was seen as the maximum detection time required in Table 4.2. This is because there is one scenario in which the lowest possible detection time is 518.5 seconds. Therefore, even the inclusion of a CVaR into the problem formulation cannot improve this.

An overhead view of the sensor placement layouts generated by Problem (SP) and Problem (SP-CVaR) is shown in Figure 4.6. Both the x and y coordinates in the facility as shown in meters. Additionally, because this is an overhead view, the z-coordinate is not shown. Therefore, sensors may lie on different vertical locations, but may appear to be on the same point. The $\times$ symbols represent all of the potential sensor locations. The sensors placed by problem (SP) are represented by the $\square$ symbols, while those placed by problem (SP-CVaR) are marked with the $*$ symbol. Of the 30 sensors placed by each formulation, at least 10% of the sensors were located in different places. Therefore, while the expected value of the detection time changes little, there are significant differences in the sensor placement found.

4.2.3 Data Set 3 Results

In Table 4.3, the results for the solution of problem formulations (SP) and (SP-CVaR) for data set 3 are shown. These problems were both solved with a maximum number of sensors p=55 to be placed. In each of the problems, all of the 145 scenarios in data set 3 were detected. This table shows the minimum, mean, and maximum detection time across all of the scenarios. Naturally, Problem (SP) yields the best expected detection time of the two formulations, with a value of 17.99 seconds for the expected detection time across all scenarios (or the mean). The worst-case detection time for Problem (SP) was higher at 26.20 seconds. Problem (SP-CVaR) was solved

with 3 different values of θ. For the previous results, θ=0.9 and θ=0.95 were also considered, but because data set 3 has much fewer scenarios, the largest value of θ considered was 0.8. The target value for CVaR was determined by solving Problem (CVaR), shown in Section 4.1, and then was used in the constraint on CVaR in Problem (SP-CVaR). Problem (SP-CVaR) has only a slightly higher expected detection time than Problem (SP), while providing a better worst-case detection time.

	Min	Mean	Max
Problem (SP)	15.00	17.99	26.20
Problem (SP-CVaR)			
θ=0.80	15.00	18.04	25.21
θ=0.20	15.00	17.99	26.20
θ=0.10	15.00	17.99	26.20

Table 4.3: Data Set 3: Minimum, maximum, and mean values for the damage coefficients of each problem formulation.

The distribution of the detection times for all scenarios is shown for each of the optimal sensor placements in Figure 4.7. The top panel shows the detection times for Problem (SP). The bulk of the events are detected towards the lower end of the distribution, however there are a significant portion of the scenarios above the 22 second mark. The mean detection time for the distribution is lower than Problem (SP-CVaR), shown in the bottom panel, but the tail end of the distribution is not as favorable. Because Problem (SP-CVaR) explicitly considers this tail of the distribution, it performs better at reducing the number of events above 22 seconds and lowers the maximum detection time. When Problem (SP-CVaR) is formulated

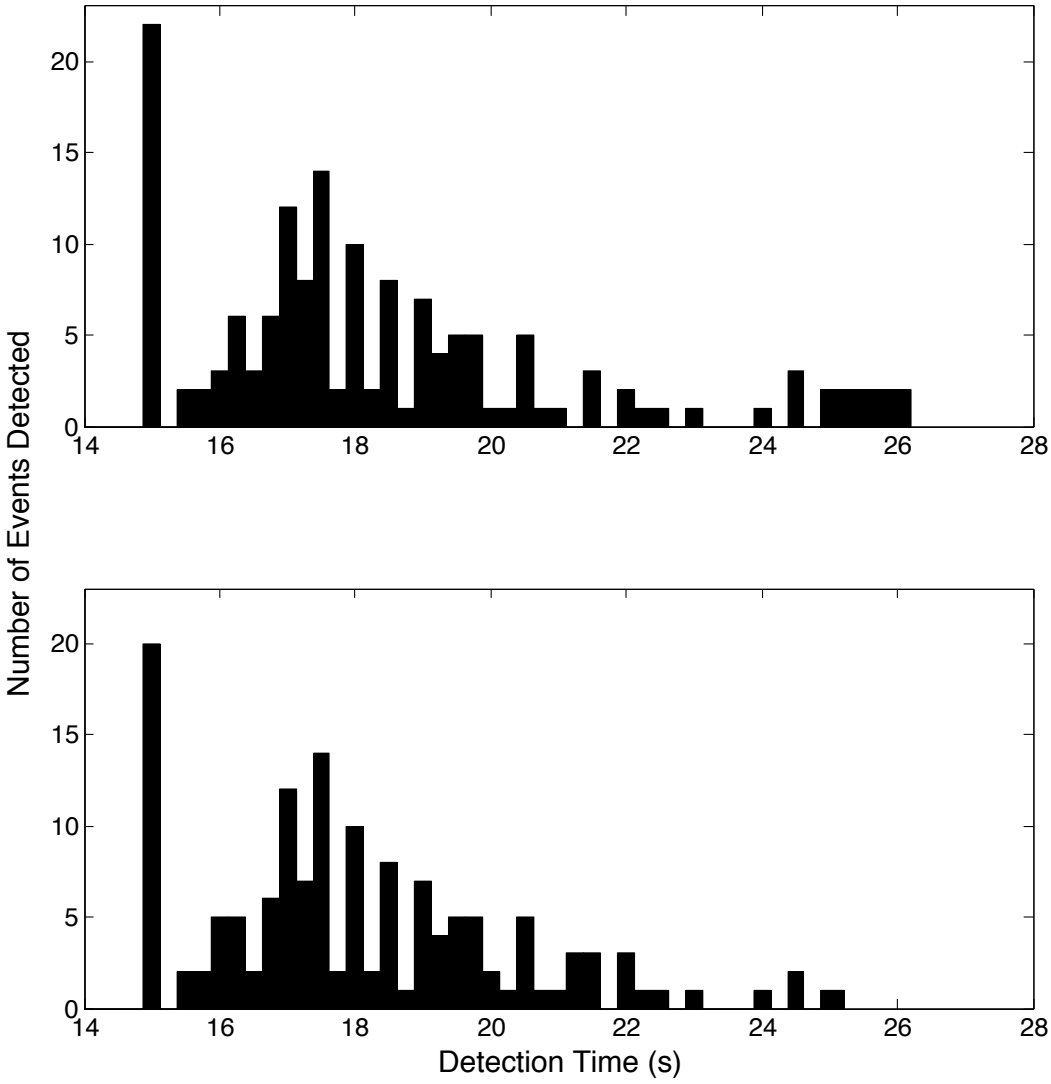

Figure 4.7: Expected detection times for (SP) and (SP-CVaR) for Data Set 3. Expected detection times for all of the scenarios in the solution of problem SP (top) and problem SP-CVaR (bottom) where θ=0.80.

with a limit on the CVaR for the 80%-level tail distribution, *i.e.* θ=0.8, roughly 45% fewer scenarios are detected above the 22 second mark. This is achieved while also sacrificing very little in the mean detection time of the distribution.

The overhead view of the sensor placement layouts generated by Problem (SP) and Problem (SP-CVaR) is shown in Figure 4.8. The locations are shown in terms of the x-y coordinates in meters. The vertical locations are not shown in this figure, therefore sensors could appear to be located at the same location while being at different vertical heights. The $\times$ symbols represent all of the potential sensor locations. The sensors placed by problem (SP) are represented by the $\square$ symbols, while those placed by problem (SP-CVaR) are marked with the $*$ symbol.

4.3 Summary

In this Chapter, the previously developed mixed-integer programming formulation for the placement of gas detectors in a petrochemical facility, Problem (SP), was extended by incorporating a constraint on the Conditional-Value-at-Risk. These formulations utilize leak scenario data from modern, rigorous CFD simulation tools. The first formulation, defined in this paper as problem formulation (SP), is designed to determine a sensor placement that minimizes the expected value of the detection time. This objective value can be extended to other desired metrics, depending on the goals of the problem and the data available. Problem (C) seeks to determine a sensor placement that minimizes the CVaR of the scenario set. This formulation is used to provide a target upper bound value for CVaR in problem formulation (SP-CVaR). Problem (SP-CVaR), like problem (SP), determines a sensor placement that minimizes the expected detection time across all scenarios, but imposes a limit on the upper bound of CVaR using the target value provided by formulation (C). Each of the problem formulations was implemented using the optimization modeling

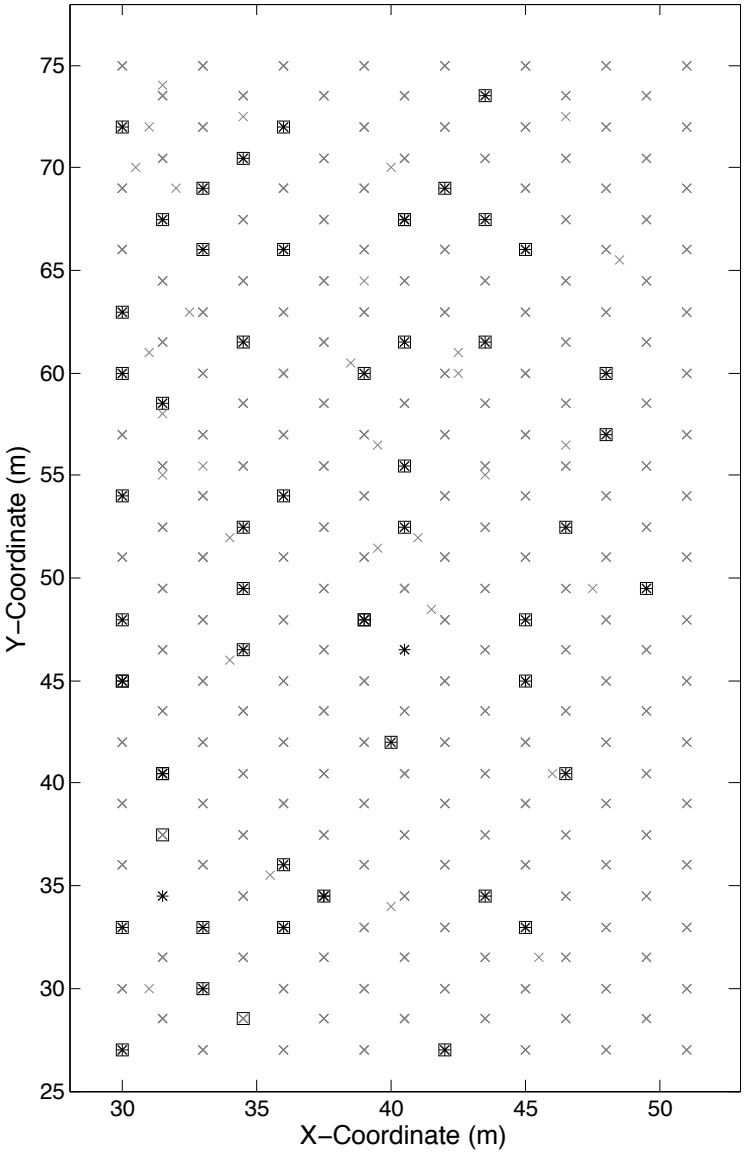

Figure 4.8: Locations of sensors placed by (SP) and (SP-CVaR) in process facility for Data Set 3. $\times$ represent potential sensor locations, $\square$ represent sensors placed by formulation (SP), and $*$ represent sensors placed by formulation (SP-CVaR). The value of θ=0.80.

framework, Pyomo. The solution of both of these problem formulations is efficient, requiring only a few seconds for Problem (SP) and less than a minute for most values of θ for Problem (SP-CVaR).

The numerical results for this chapter were obtained utilizing the three data sets provided by GexCon, each representing a realistic process facility. In each of the results, Problem (SP) provided a sensor placement with good performance in minimizing the expected detection time across all scenarios. As expected, this formulation provided the lowest mean detection time. Additionally, it produced a distribution of detection times with a bulk of the events requiring little time to detect. However, this formulation produced a placement that gives a wide distribution of detection times and does not have good worst-case performance. For data set 1, the maximum detection time with this placement was 110.5 seconds, while 12 scenarios required more than 65 seconds to detect. Given these results, a strong case was made to extend this formulation to improve the worst-case behavior. For data set 2, the maximum detection time was 518.5 seconds, while 8 scenarios required more than 18 seconds to detect. Finally, for data set 3, the maximum detection time for Problem (SP) was 26.20 seconds, while 19 scenarios required more than 22 seconds to detect.

Problem (SPC) incorporated a restriction on the CVaR value of the scenario set into the original problem formulation (SP). The CVaR upper bound is determined through a minimization of the value of CVaR, shown in Section 4.1 as problem formulation (CVaR). Minimum, mean, and maximum detection times for the scenario set were tabulated for problem (SPC) for several values of the θ-level distribution for each data set. Problem (SP-CVaR) showed a slight increase in the expected detection time across all scenarios compared to problem (SP), but provided a sensor placement that reduced the worst-case detection time by almost 30 seconds for data set 1, no reduction in the worst-case time for data set 2 because one scenario could

only be detected in a very large time span, and a reduction of 1.0 seconds for data set 3. Additionally, for data set 1, only 6 scenarios required more than 65 seconds to detect, which is half that of problem (SP). For data set 2, only 5 scenarios required more than 18 seconds to detected (overall reduction of more than 35%), and for data set 3, only 10 scenarios required more than 22 seconds to detect. Furthermore, the results for data set 1 show that both of these optimization-based approaches produced sensor placements that provide significantly better performance than that based on coverage alone.

These results show that a formulation designed to limit the worse-case scenarios while also providing good performance on the mean of the distribution of detection times for all events is possible. This is important because the previous formulations showed great performance on minimizing the mean detection time across all scenarios, but still had several worst-case scenarios. These worst-case scenarios may lead to very high impact events, and need to be mitigated as much as possible. The SP-CVaR formulation reduces the impact of these scenarios while also providing a good mean detection time. Because this formulation is still very computationally efficient, it can be solved on standard computers, making it a tool available to designers of safety systems without the need for supercomputers.

5. DETERMINING THE QUALITY OF DETECTOR PLACEMENTS*

A major challenge in stochastic programming is deciding whether or not enough scenarios were used to adequately represent the uncertainty space. There are two related aspects to consider. First, given an optimal placement found using one set of leak scenarios, we wish to evaluate the performance of this candidate placement on alternate sets of leak scenarios. This is done using Monte-Carlo subsampling of our scenario set and provides us some measure of the reliability of the candidate solution to unforeseen leak scenarios. Second, given a candidate gas detector placement, we would like to know confidence intervals on the value of the objective if we could consider the entire uncertainty space. This is calculated using the procedure of Mak et al. (1999) to determine the 95% confidence intervals on the optimality gap statistic.

5.1 Implementation of the Procedure of Mak, Morton, and Wood [1999]

The procedure combining these two aspects is outlined as follows:

1. n scenarios are randomly sampled from the full scenario set to generate a candidate sensor placement.

2. An optimal placement given these scenarios is determined using p sensors. This candidate placement is stored in $\hat{s}$.

3. For each $i = 1, ..., n_g$:

   (a) n scenarios are again randomly sampled from the full scenario set.

*Part of this section is reprinted with permission from "A Stochastic Programming Approach For Gas Detector Placement Using CFD-based Dispersion Simulations" by Legg SW, Benavides-Serrano AJ, Siirola JD, Watson JP, Davis SG, Bratteteig A, and Laird CD, 2012. Computers & Chemical Engineering, 47(0):194-201, Copyright 2012 by Elsevier Ltd.

(b) The optimization formulation considering these scenarios is solved, and the optimal objective value is stored as F_i^*.

(c) Using these same scenarios, the objective function is evaluated with fixed $s{=}\hat{s}$. The objective value resulting from this candidate placement is stored as F_i^c.

4. A histogram of all values in F^c is generated. This provides statistics regarding the effectiveness of the candidate placement $\hat{s}$ on scenario sets that were not considered when generating the candidate solution.

5. Optimality gap statistics are then computed according to Mak et al. (1999), where

$$G_i{=}F_i^c{-}F_i^*,$$
$$\bar{G}{=}\frac{1}{n_g}\sum_{i=1}^{n_g} G_i,$$
$$s_G{=}\sqrt{\frac{1}{n_g{-}1}\sum_{i=1}^{n_g}(G_i-\bar{G})^2}$$

and the approximate $(1{-}\alpha)$-level confidence interval for the optimality gap is

$$\left[0, \bar{G}{+}\frac{t_{\alpha,n_g-1}\cdot s_G}{\sqrt{n_g}}\right].$$

Here, n_g represents the number of subsampled scenario sets to consider, while n is the number of scenarios included in each subsample. The above procedure is used

to evaluate the performance of the candidate placements using each of the three placement strategies described previously: minimizing expected detection time (SP), minimizing expected detection time while satisfying a constraint on coverage (SPC), and coverage-only (C). The coverage-only based placement does not optimize over any scenarios, therefore it is only reasonable to generate the histograms from step 4, and not the optimality gap statistics. The number of subsampled scenario sets considered was n_g=1000, and the number of scenarios in each subsample is n=75. Since the number of available scenarios (275 for Data Set 1, 314 for Data Set 2, 145 for Data Set 3) is not significantly larger than the subsample size (75), we were not able to perform random partitioning of the full scenario space. Rather, subsamples were randomly drawn and replaced before the next set of subsamples were drawn. This means that some scenarios may be shared across different subsample sets, and the reported results may have smaller variance than that which would be produced if there were enough scenarios for full partitioning. Sampling without replacement is preferred, so the use of more scenarios is recommended if available.

5.2 Numerical Results

5.2.1 Data Set 1 Results

Considering problem (SP) allowing 50 sensors (p=50), repeatedly optimizing over n=75 scenarios a total of n_g=1000 times produces a mean optimal value of $\bar{F}^*$=25.80 seconds with a standard deviation of only 0.402 seconds. Furthermore, for each of the 1000 optimizations, all leak scenarios were detected. These values can be seen in Table 5.1. This shows that this approach is able to effectively place 50 sensors to protect against any one set of 75 scenarios. However, we are more interested in the effectiveness of a candidate placement on unforeseen scenarios that were not considered during the optimization. Figure 5.1 shows histograms of F^c (from step

4) for candidate placements from each of the three approaches. Recall that these values are calculated using the candidate placement ($\hat{s}$) from step 2 evaluated over 1000 randomly selected subsets of 75 scenarios each. The mean expected detection time for the candidate placement from problem (SP) is 60.6 seconds with a standard deviation of 9.5 seconds. The top panel in 5.1 shows the complete histogram of F^c for problem (SP). Furthermore, on average, the fraction of scenarios detected by the candidate placement was 0.94 with a standard deviation of 0.02, with the histogram of the fraction of scenarios detected shown in the top panel of Figure 5.2. While problem (SP) produces a good optimal objective value when determining the candidate placement, when this placement ($\hat{s}$) is evaluated on alternate scenario sets, the performance decreases significantly.

We expect that problem formulation (SPC) will produce a candidate placement that is more reliable on alternate scenarios. The mean expected value $\bar{F}^*$ for problem (SPC) is 29.30 with a standard deviation of 0.735, slightly worse than that from problem (SP). However, the mean expected time for the candidate placement on the alternate scenario sets ($\bar{F}^c$) is 37.03 seconds with a standard deviation of only 3.4 seconds. The mean fraction of scenarios detected was 0.994 with a standard deviation of 0.007, a considerable improvement over the candidate placement from (SP). As expected, problem formulation (SPC) produces a candidate placement that performs significantly better on alternate scenario sets than that produced by problem (SP). Full histograms of both F^c and the fraction of scenarios detected for the candidate placement from (SPC) are shown in the middle panels of Figures 5.1 and 5.2, respectively.

The bottom panels in Figures 5.1 and 5.2 show the values for F^c and the fraction of scenarios detected by the candidate placement from (C). The mean expected detection time $\bar{F}^c$ was 150.1 seconds with a standard deviation of 15.12 seconds,

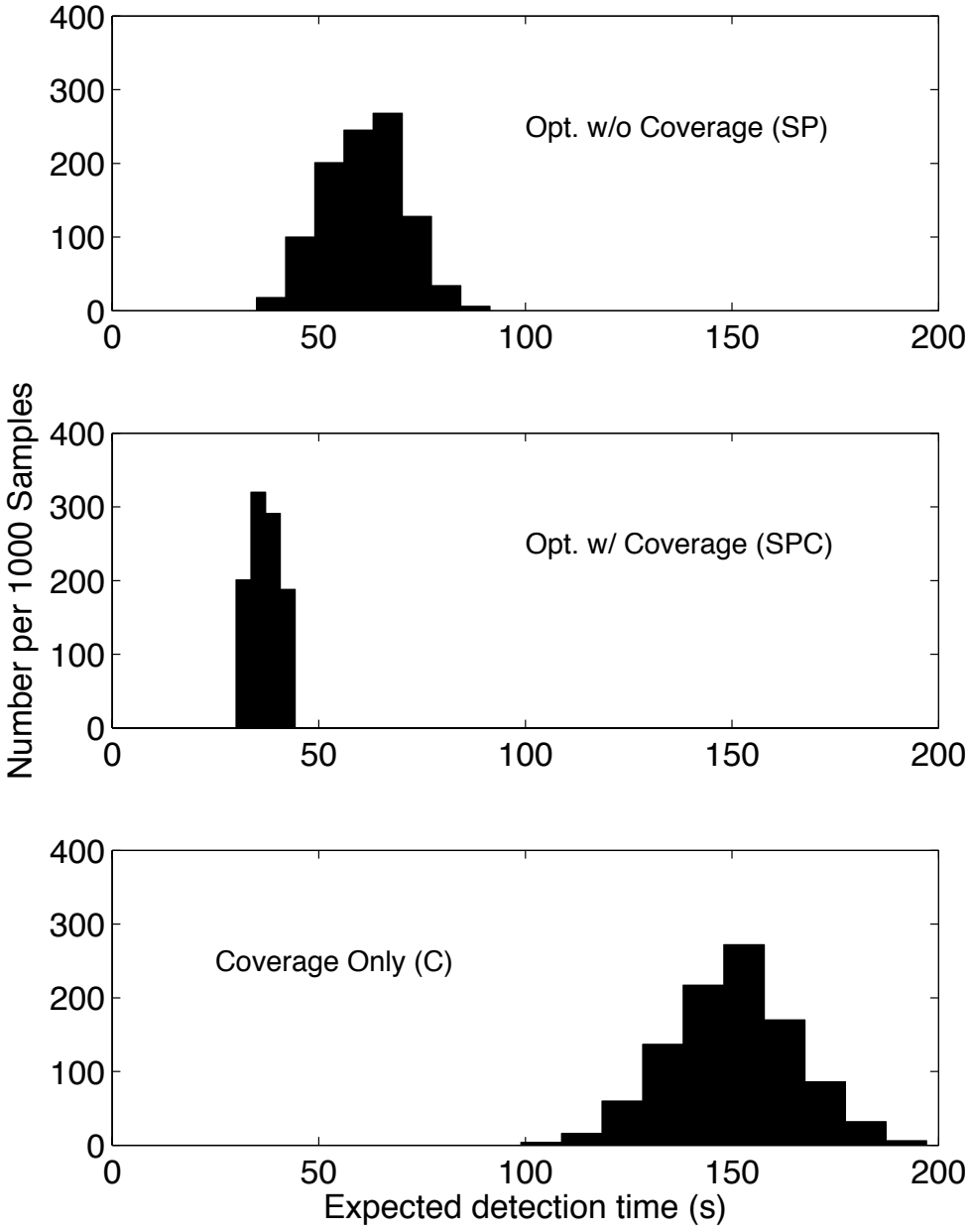

Figure 5.1: Data Set 1: Expected detection times F_i^c for candidate placements from different approaches

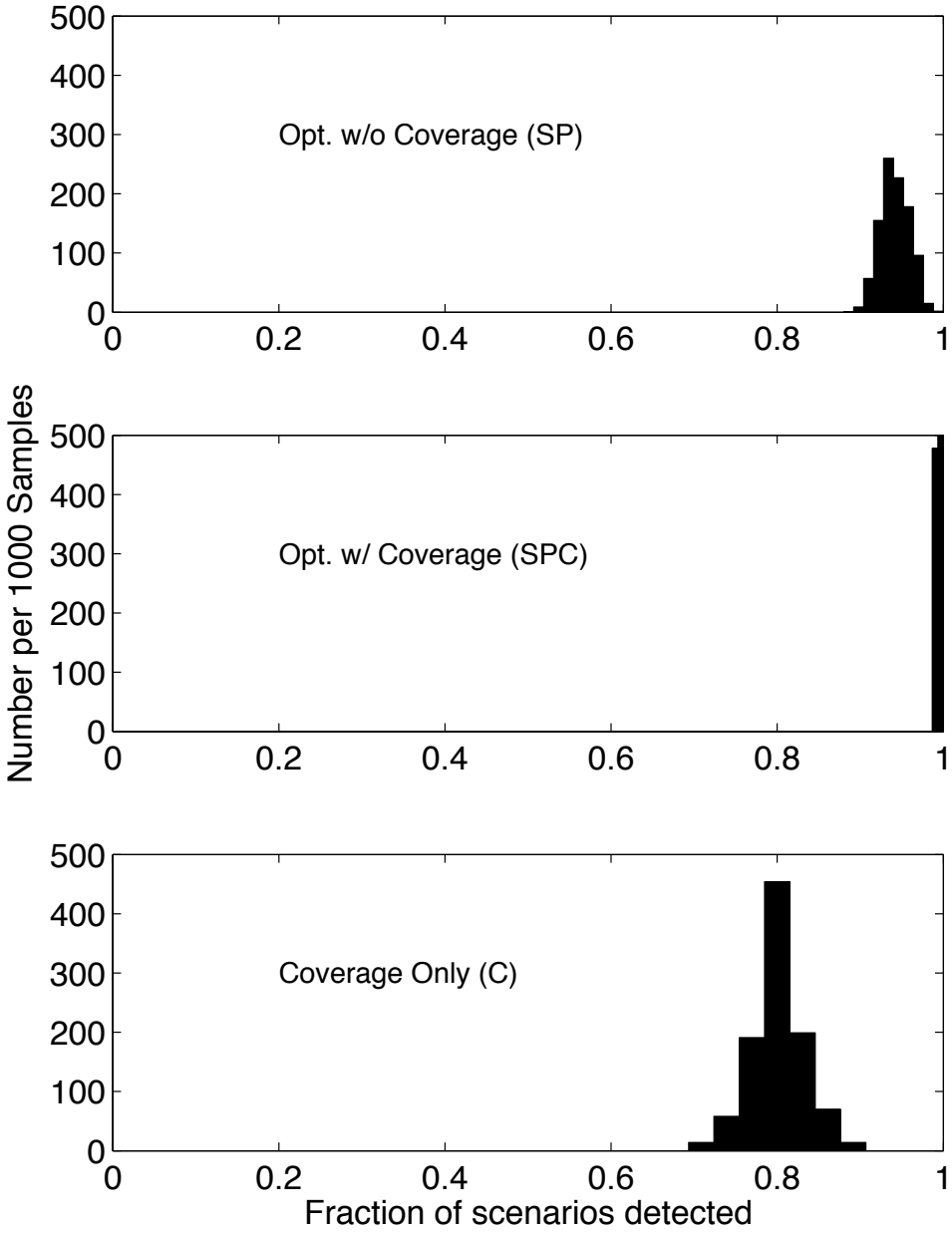

Figure 5.2: Data Set 1: Fraction of scenarios detected for candidate placements from different approaches

and the mean fraction of scenarios detected is 0.8 with a standard deviation of 0.03. These figures show that the candidate placement based on coverage alone performs significantly worse than both optimization based approaches. Tables 5.1 and 5.3 show a summary of the statistics for each of the three placement formulations.

	(SP)	(SPC)
Mean Exp. Time (s)	25.80	29.30
St.Dev. Exp. Time (s)	0.402	0.735
Mean Fraction Detected	1.000	1.000
St.Dev. Fraction Detected	0.000	0.000

Table 5.1: Data Set 1: Statistics for F^* from Step 3.2

While it is common to approximate the full uncertainty space with a finite number of scenarios, because of the high computational cost associated with rigorous CFD simulation of the leak scenarios, scenario data is likely to be limited. Given a candidate solution, the procedure of Mak et al. (1999) provides confidence intervals on the optimality gap if this candidate solution was applied over a much larger set of scenarios (i.e. so large that the full uncertainty space is fully approximated). Confidence intervals were calculated using n_g=30 samples. The results presented are the worst optimality gap for ten different runs of the sampling procedure.

Table 5.2 includes the 95% confidence intervals on this optimality gap for both formulations (SP) and (SPC). As expected, while formulation (SP) produces the lowest mean objective value ($\bar{F}^*$), the confidence intervals on the optimality gap are large. However, while formulation (SPC) has a slightly worse mean objective value

($\bar{F}^*$), the 95% confidence intervals are much smaller, indicating that this placement may outperform the placement from (SP) over the entire uncertainty space. The 95% confidence intervals on the optimality gap cannot be determined for this coverage only placement because there is no "optimal" placement given a set of scenarios.

	(SP)	(SPC)	(C)
Mean Exp. Time (s)	60.63	37.03	150.1
St.Dev. Exp. Time (s)	9.532	3.445	15.12
Mean Fraction Detected	0.941	0.994	0.801
St. Dev. Fraction Detected	0.020	0.007	0.033
95% Confidence Interval Optimality Gap Statistic	[0, 38.94]	[0, 10.12]	—

Table 5.2: Data Set 1: Statistics for F^c from Steps 3.3 and 5

5.2.2 Data Set 2 Results

As with the results from data set 1, repeatedly optimizing problem (SP) over n=75 scenarios a total of n_g=1000 times produces a mean optimal value of $\bar{F}^*$=17.13 seconds with a standard deviation of only 2.908 seconds for data set 2. All leak scenarios were detected for each of the 1000 subsamples. These results show that this approach effectively protects against any set of 75 scenarios by placing 30 sensors. Next, we would like to analyze how this approach performs in the face of unforeseen scenarios that were not used during the optimization. In Figure 5.3, histograms of F^c (from step 4) for candidate placements are shown for the three approaches (SP, SPC, and C). These values are determined by evaluating the placement from step 2 ($\hat{s}$) over

the 1000 random subsets of 75 scenarios. The mean expected detection time for the candidate placement from problem (SP) is 34.36 seconds, with a standard deviation of 11.94 seconds. The complete histogram of the expected detection times for each of the random samples is shown in the top panel of Figure 5.3. Additionally, the mean fraction of events detected by the candidate placement is 0.983, with a standard deviation of 0.013. The top panel of Figure 5.4 shows the histogram of fraction of events detected for problem (SP). While the placement generated by problem (SP) provides a very good optimal objective value when solving for a candidate placement, it is apparent that the performance suffers when this placement is evaluated on alternate scenario sets. Table 5.3 shows a summary of these results.

As postulated in the results for data set 1, it is expected that problem (SPC) will produce a more reliable candidate placement for alternate scenario sets. For problem (SPC), the mean expected value $\bar{F}^*$=17.91, with a standard deviation of 2.957. These values are slightly worse than those produce by problem (SP). Again, all of the events were detected. When evaluated on alternate scenario sets, the mean expected detection time for problem (SPC), $\bar{F}^c$, was 27.42 with a standard deviation of 7.916. This is a noticeable improvement over the performance of problem (SP). The mean fraction of events detected is 0.997 with a standard deviation of 0.005, also showing improvement over problem (SP). Histograms for F^c and the fraction of scenarios detected for the candidate placement from problem (SPC) are shown in the middle panels of Figures 5.3 and 5.4, respectively.

In order to compare the performance of problems (SP) and (SPC) to the coverage-only approach (C), the histograms for the expected detection times and fraction of events detected for placement (C) are shown in the bottom panels of Figures 5.3 and 5.4, respectively. The mean expected detection time for all samples ($\bar{F}^c$) is 44.61 seconds with a standard deviation of 13.40 seconds. The mean fraction of scenarios

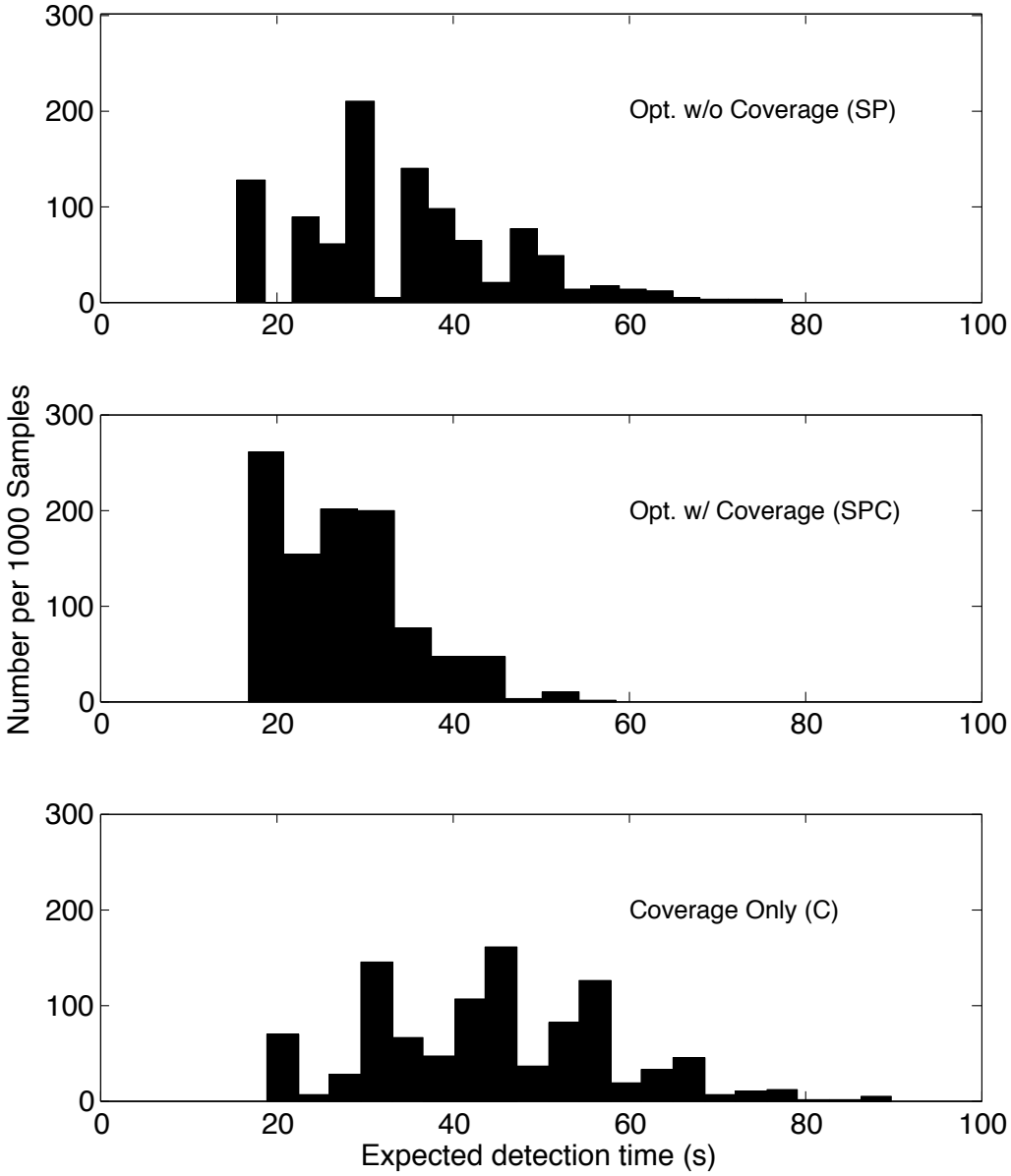

Figure 5.3: Data Set 2: Expected detection times F_i^c for candidate placements from different approaches.

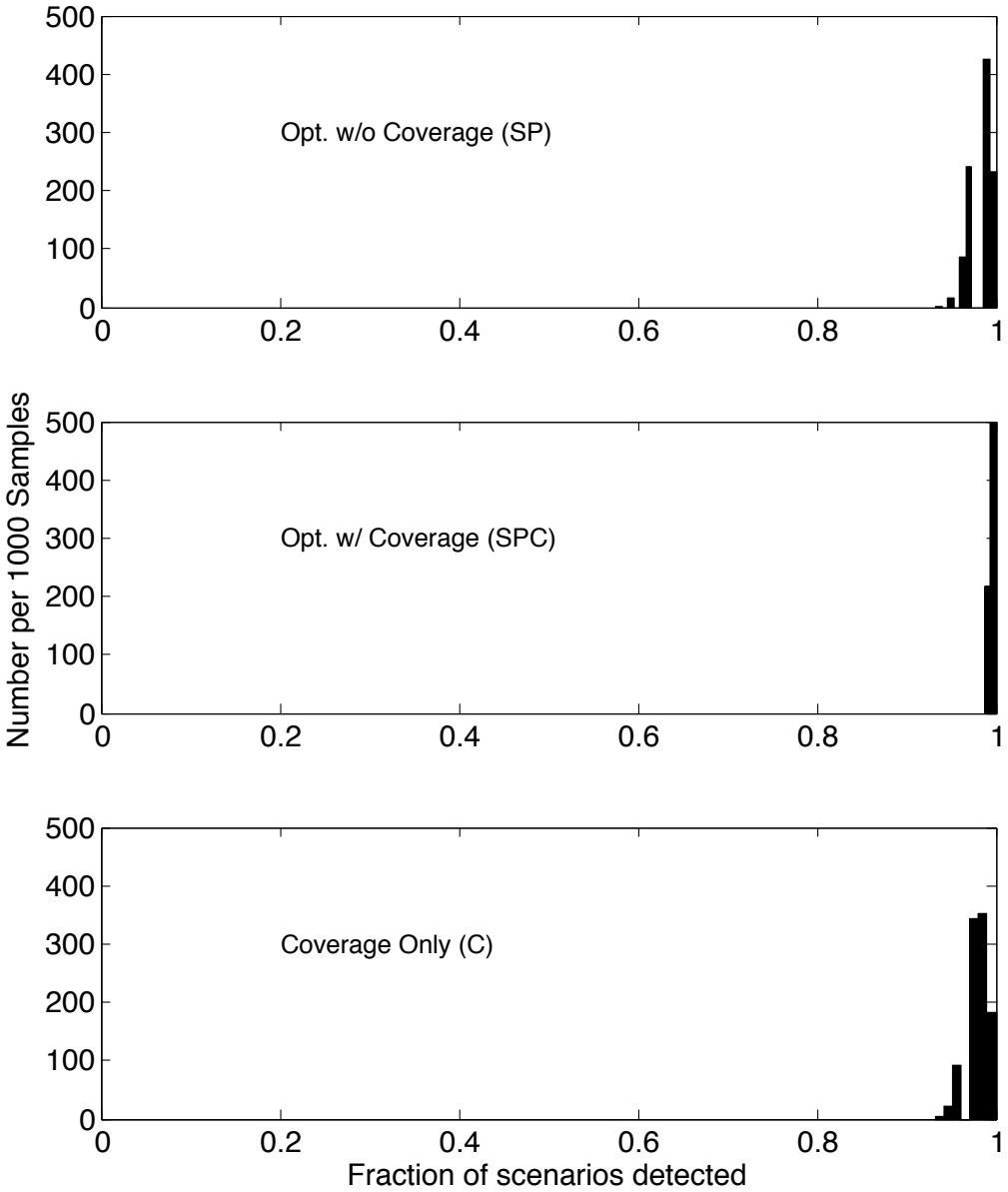

Figure 5.4: Data Set 2: Fraction of scenarios detected for candidate placements from different approaches.

detected is 0.980 with a standard deviation of 0.013. Again, the optimization-based approach outperforms the pure coverage-based approach on this scenario set.

The 95% confidence intervals on the optimality gap for both formulations (SP) and (SPC) are shown in Table 5.4. Formulation (SPC) outperforms formulation (SP) when candidate solutions are evaluated on alternate scenario sets. The expected objective is lower, and the confidence intervals are tighter. This indicates that the placement generated by problem formulation (SPC) should outperform the placement generated by problem (SP) over the entire uncertainty space. There is no 95% confidence interval for the coverage-only placement because there is no "optimal" coverage-only placement for a particular set of scenarios.

	(SP)	(SPC)
Mean Exp. Time (s)	17.13	17.91
St.Dev. Exp. Time (s)	2.908	2.957
Mean Fraction Detected	1.000	1.000
St.Dev. Fraction Detected	0.000	0.000

Table 5.3: Data Set 2: Statistics for F^* from Step 3.2

5.2.3 Data Set 3 Results

For data set 3, repeatedly optimizing problem (SP) over n=75 scenarios a total of n_g=1000 times produced a mean optimal value of $\bar{F}^*$=17.43 seconds with a standard deviation of 0.153 seconds. Furthermore, all leak scenarios were detected for each of the samples. These results are summarized in Table 5.5. By placing 55 sensors, problem formulation (SP) effectively protects against any set of 75 scenarios from

	(SP)	(SPC)	(C)
Mean Exp. Time (s)	34.36	27.42	44.61
St.Dev. Exp. Time (s)	11.94	7.916	13.40
Mean Fraction Detected	0.983	0.997	0.980
St. Dev. Fraction Detected	0.013	0.005	0.013
95% Confidence Interval Optimality Gap Statistic	[0, 17.83]	[0, 9.881]	—

Table 5.4: Data Set 2: Statistics for F^c from Steps 3.3 and 5

the full scenario set. However, when faced with scenarios not in the set used to optimize the placement, the performance suffers. The performance of the placement on unforeseen scenarios is determined by evaluating the placement ($\hat{s}$) from step 2 in Section 5.1 over 1000 random subsets of 75 scenarios each. The top panels of Figures 5.5 and 5.6 show the histograms for the expected detection time and fraction of scenarios detected by the candidate placement from (SP) on alternate subsamples of scenarios. The mean expected detection time for all samples is 45.01 seconds with a standard deviation of 8.260 seconds. The mean fraction of events detected is 0.951 with a standard deviation of 0.0167. While problem formulation (SP) performs well on the set of scenarios used to optimize the placement, the performance of the placement suffers when faced with unforeseen scenarios.

For problem (SPC), the mean optimal value of $\bar{F}^*$=18.19 seconds with a standard deviation of 0.246 seconds, slightly worse than that produced by problem (SP). All scenarios were detected. When evaluated on alternate samples of scenarios, problem (SPC) yielded a mean expected detection time $\bar{F}^c$=24.47 seconds with a standard deviation of 3.295 seconds. The fraction of events detected is 0.993 with a standard

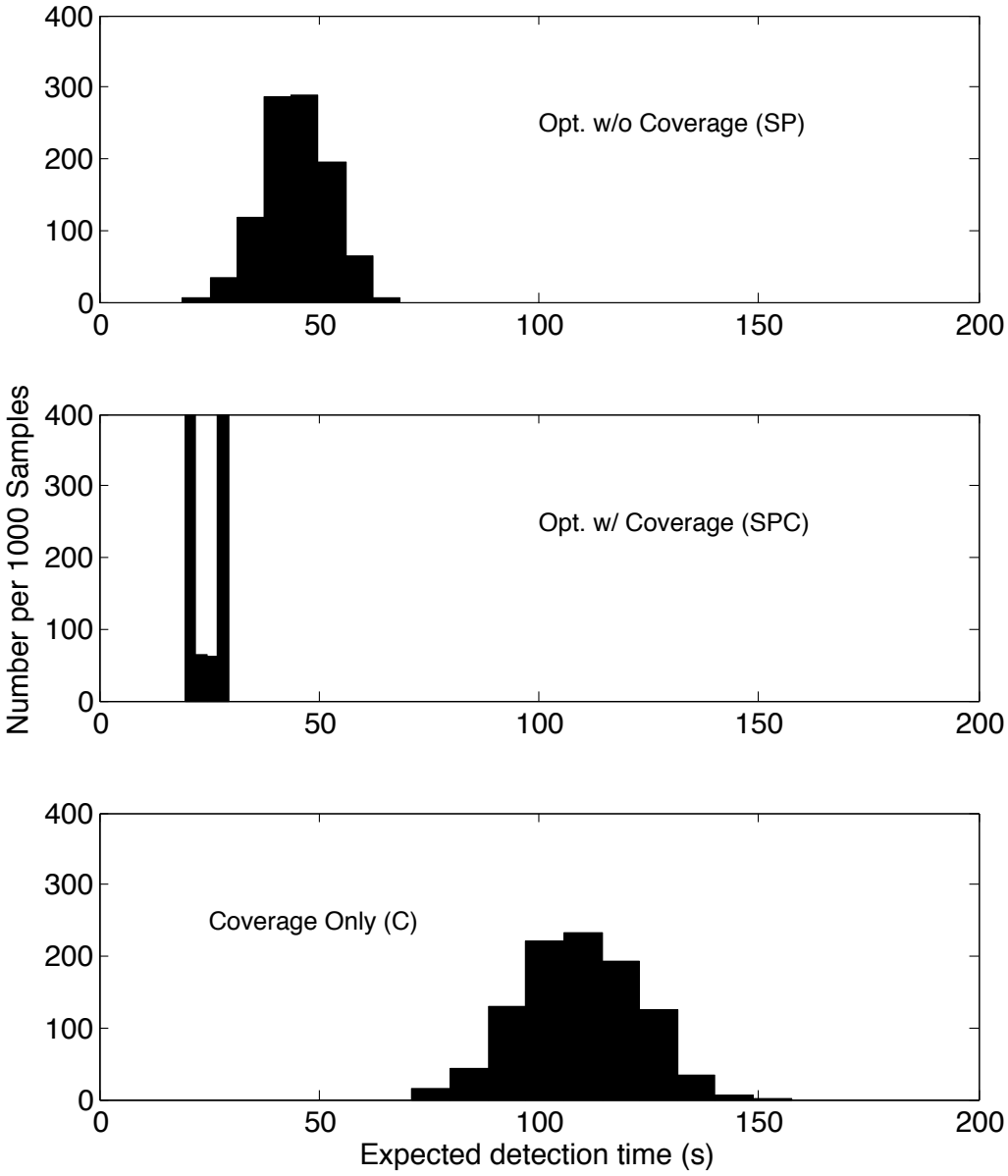

Figure 5.5: Data Set 3: Expected detection times F_i^c for candidate placements from different approaches.

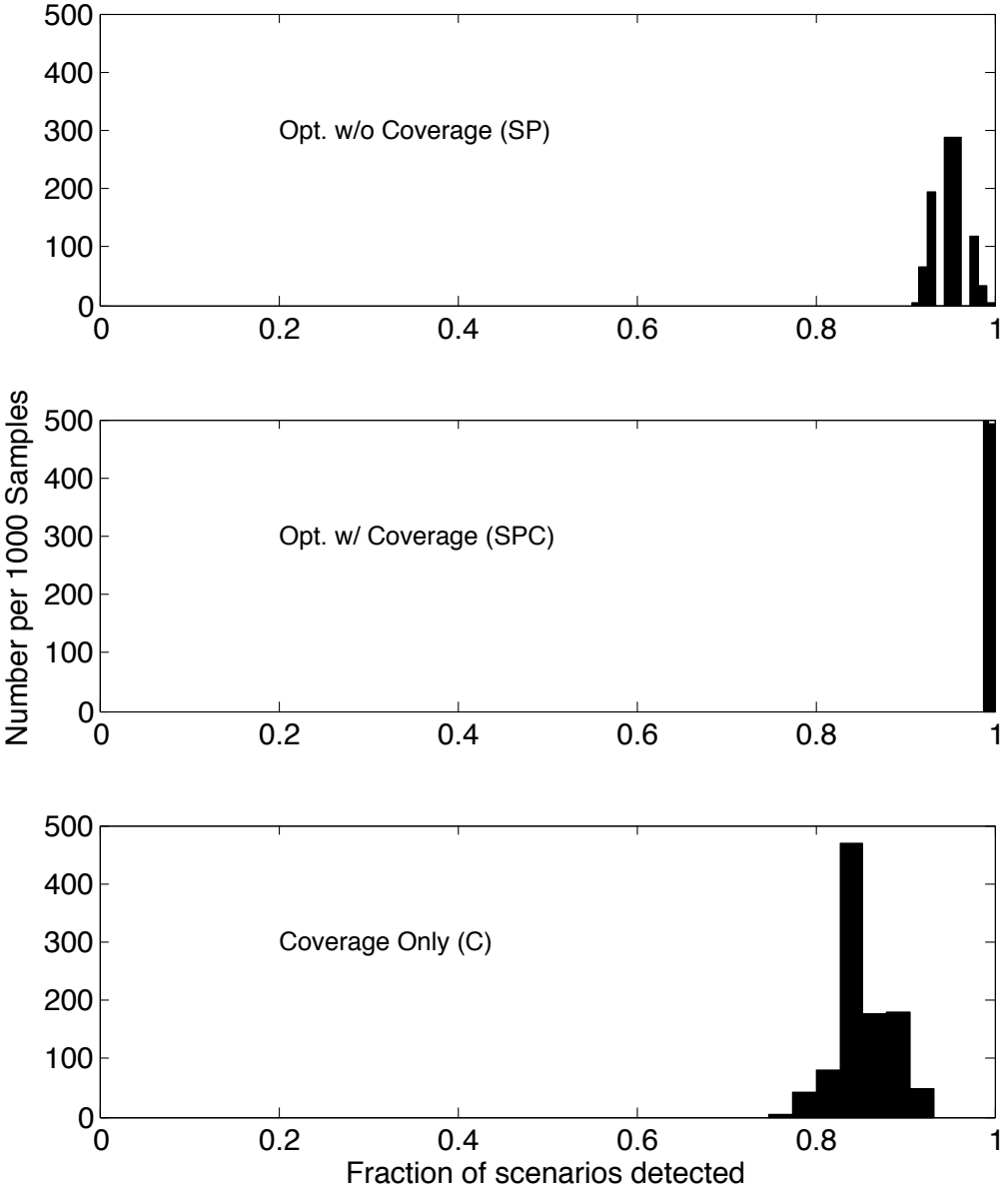

Figure 5.6: Data Set 3: Fraction of scenarios detected for candidate placements from different approaches.

deviation of 0.007, a marked improvement over the placement generated by problem (SP). The histograms for F^c and the fraction of scenarios detected for problem (SPC) are shown in the middle panels of Figures 5.5 and 5.6, respectively.

For the coverage-only approach (C), the expected detection times and fraction of scenarios detected are shown in the bottom panels of Figures 5.5 and 5.6, respectively. Placement (C) produced a mean expected detection time $\bar{F}^c$=109.0 seconds with a standard deviation of 13.45 seconds, and the mean fraction of scenarios detected is 0.853 with a standard deviation of 0.028. For this data set, the coverage-only placement performs significantly worse than either of the two optimization based approaches.

Table 5.6 shows the 95% confidence intervals on the optimality gaps for problem formulations (SP) and (SPC). While problem (SP) produces the lower F^* of the two formulations, its expected objective value is worse, and it also produces a much larger confidence interval on the optimality gap. The difference in the optimality gaps between formulations (SP) and (SPC) shows that the placement generated by problem (SPC) should perform much better over the full uncertainty space.

	(SP)	(SPC)
Mean Exp. Time (s)	17.43	18.19
St.Dev. Exp. Time (s)	0.153	0.246
Mean Fraction Detected	1.000	1.000
St.Dev. Fraction Detected	0.000	0.000

Table 5.5: Data Set 3: Statistics for F^* from Step 3.2

	(SP)	(SPC)	(C)
Mean Exp. Time (s)	45.01	24.47	109.0
St.Dev. Exp. Time (s)	8.260	3.295	13.45
Mean Fraction Detected	0.951	0.993	0.853
St. Dev. Fraction Detected	0.0167	0.007	0.028
95% Confidence Interval Optimality Gap Statistic	[0, 28.01]	[0, 6.452]	—

Table 5.6: Data Set 3: Statistics for F^c from Steps 3.3 and 5

5.3 Summary

In this chapter, previously developed mixed-integer approaches for optimal placement of gas detectors in a petrochemical facility were assessed on scenarios not known during the optimization. By far, the dominant computational cost in the entire approach to optimally determining sensor placements is the simulation of leak scenarios. Simulation of a single leak scenario can take hours to days of computational time. Therefore, there is a desire to limit the number of required scenarios. However, there is no guarantee that a small number of scenarios will adequately approximate the uncertainty space (leak location, process conditions, weather, etc.). We have outlined a sampling procedure based on the work of Mak et al. (1999) that quantifies the effectiveness of candidate placements on alternate scenario sets not considered in the optimization. This procedure also provides confidence intervals on the optimality gap for a candidate placement over the entire uncertainty space. Note that this sampling procedure uses subsamples from a predetermined fixed set of scenarios. If the original set of scenarios is somehow biased, the calculated statistics may not be sufficiently conservative, leading to false confidence in candidate sensor placements.

Therefore, it is important that the simulated leak scenarios truly represent a random sampling from the uncertainty space. All formulations analyzed in this chapter were compared on three independent data sets provided by GexCon.

Formulation (SP) generates sensor placements with good performance over the set of scenarios considered. We recommend this problem formulation if the scenario set is large enough to produce sufficiently small confidence intervals on the optimality gap statistic. A procedure such as that described in Section 5.1 should be used to determine if enough scenarios are available to adequately represent the uncertainty space. If there are not enough scenarios to provide the necessary performance guarantees, there are two alternatives. We can either perform additional CFD simulations, and increase the number of scenarios, or we can modify the problem formulation to provide improved reliability.

Given the high cost of simulating scenarios using CFD modeling, it is more desirable to improve the problem formulation. Problem formulation (SPC) includes constraints on the overall volume coverage. This hybridized approach meets the coverage goals while still making effective use of the rigorous simulation data for improved performance. Problem formulation (SPC) sacrifices little in terms of the optimal objective value over the scenarios considered, but provides significantly improved detection of scenarios that were not considered in the optimization. Using formulation (SPC), the mean value $\bar{F}^c$ was significantly better than that obtained using candidate solutions from formulation (SP). Therefore, if the number of scenarios do not provide the necessary performance guarantees, we recommend using problem formulation (SPC).

It should also be noted that the performance of these two optimization-based approaches were compared with the performance from a coverage-only approach. While approaches based on coverage alone are common, for the scenarios considered in this

paper, the coverage-only placement (C) produced significantly worse performance in both the expected detection time and the fraction of scenarios detected on two of the three data sets, and performed slightly worse on a data set representing a much smaller process module. This is especially surprising since placement (C) produced a tighter coverage than was required in formulation (SPC) for the same number of sensors. This result provides significant evidence that using rigorous plant-specific CFD simulations of leak scenarios provides an effective means to both assess gas detector placement, and to perform optimal gas detector placement.

6. SCENARIO SET SIZE AND SOLUTION VARIABILITY

In Chapter 5, we showed a method for calculating the confidence interval on the full space objective function for a sensor placement determined using only a limited sample of release scenarios. This method allows us to determine how confident we are that a particular sensor placement will protect against scenarios across the full uncertainty space, not just those that were used to generate the placement. Even if the objective function value of two solutions is similar, however, there is no guarantee that these sensor placements are similar, and it is important to determine how different any set of proposed sensor placements may be. Here, we design a metric that determines the solution variability, or the *distance*, between any two placements. We will present a method for determining the solution variability, and demonstrate the use of this method on a test case problem similar to the gas detector placement problem.

6.1 Defining Solution Variability

Given two optimal solutions, we can define a distance metric by computing the amount of change necessary to translate one solution to the other. For solutions coded as strings, this is accomplished with an edit distance (Levenshtein, 1966; Yujian and Bo, 2007), and in the simplest forms, can be performed by counting the number of locations that differ between the two solutions. This is known as the Hamming distance (Hamming, 1950). For the gas detector placement problem, it is more appropriate to determine the distance between the two placements by some distance metric, such as Euclidean distance.

6.1.1 Assignment Problem

In order to determine the minimum distance between any two placements, we formulate the following assignment problem. We wish to assign each sensor location in one placement to a sensor in the other placement in such as fashion as to determine the minimum amount of movement necessary to translate one placement to the other. The formulation for determining the minimum total distance $D(I, J)$ between placement I and placement J is:

$$D(I,J) = \min \quad \sum_{i \in I} \sum_{j \in J} \delta_{ij} d(i,j) \tag{6.1a}$$

s.t.

$$\sum_{i \in I} \delta_{ij} = 1 \qquad \forall\, j \in J \tag{6.1b}$$

$$\sum_{j \in J} \delta_{ij} = 1 \qquad \forall\, i \in I \tag{6.1c}$$

$$\delta_{ij} \geq 0 \qquad \forall\, i \in I,\, j \in J \tag{6.1d}$$

where $d(i, j)$ is the distance between a point $i \in I$ and a point $j \in J$, $D(I, J)$ is the total distance between placements I and J, and $\delta_{i,j}$ is an assignment variable. For the gas detector problem, it is most appropriate for $d(i, j)$ to be a Euclidean distance, i.e. the "shortest-path", but other metrics may be used.

6.2 Numerical Results

For these results, it was important to have access to large scenario sets to show how increasing scenario size affects solution variability. The available CFD scenario sets used in previous chapters were not large enough for this task, so to demonstrate this approach, the municipal water network problem was used. The scenarios for

the problem can be generated extremely quickly, allowing for tens of thousands of scenarios to be generated. This problem is very similar to the gas detector problem, the primary difference is how the scenarios are generated. Because of the time discretization for the simulations, the full scenario set can be generated by simulating contamination events at ever time step at every possible location. This allows us to determine the performance of a candidate sensor placement on the "full" uncertainty space.

6.2.1 Solution Variability Results

For this case study, the water network had $3,358$ potential locations and 96 time steps, giving a total of $332,368$ scenarios. Here we will analyze the how sets of candidate solutions vary when generated with different scenario sets and how this variability decreases as more scenarios are used in each sample. The figures shown here will describe the difference between any two placements in terms of the average number of nodes hops required to translate one placement to another. This metric makes sense for the water network, because the fluids flow through a pipeline. If this were the gas detector problem, a distance metric such as euclidean distance may be more appropriate. In Figure 6.1, we see the distribution of the average number of node hops between the candidate sensor placement and each optimal trial placement when using a scenario sample size of 100 for each placement. In each of the figures presented here, the x-axis is the average number of node hops per sensor needed to translate the candidate solution to any trial solution, and the y-axis is the frequency of that number for 100 trials. The average number of hops per sensor required to translate the candidate placement to the trial placement ranges from 10 to 26, with a mean around 18. In Figure 6.2, the same result is shown, except this time the sample size of scenarios is 500. When this many samples are used, we can see that

the distribution has grown much tighter. The average number of node hops needed per sensor placed ranges only from 6 to 16.

In Figure 6.3, the number of scenarios used in each sample is increased to 1000, and again the distribution of the nodes hops becomes tighter. However, the decrease is much less than that when 500 scenarios are used for each sample. Figure 6.4 shows the results when 1500 scenarios are used. In this case, there is very little improvement in the distribution over the 1000 sample case. This means at this point using more scenarios in the samples used to generate the sensor placement is showing little improvement. By looking at these 4 plots, one can judge how the much benefit increasing sample sizes may bring. In this example there was little improvement in the variance beyond 1000 scenarios. This is important since the full scenario set contains over $300,000$ scenarios. Using a distance measure to determine the variability between two placements can be a useful tool for determining when a sufficiently large sample of scenarios is being used to determine an optimal placement.

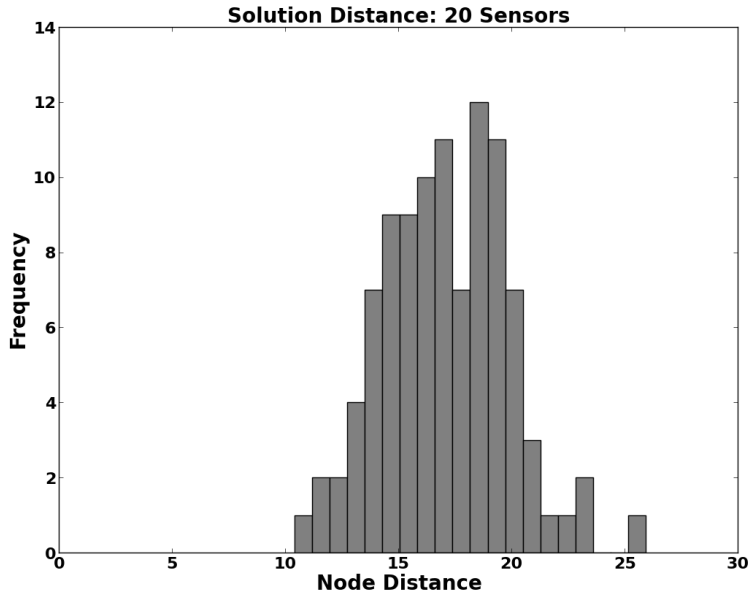

Figure 6.1: Distribution of average number of node hops per sensor to translate candidate solution to trial solutions for 100 scenarios.

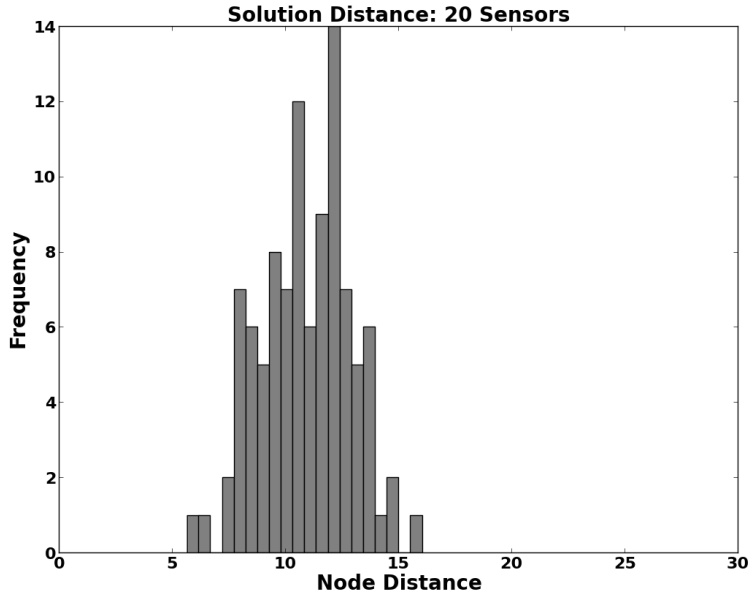

Figure 6.2: Distribution of average number of node hops per sensor to translate candidate solution to trial solutions for 500 scenarios.

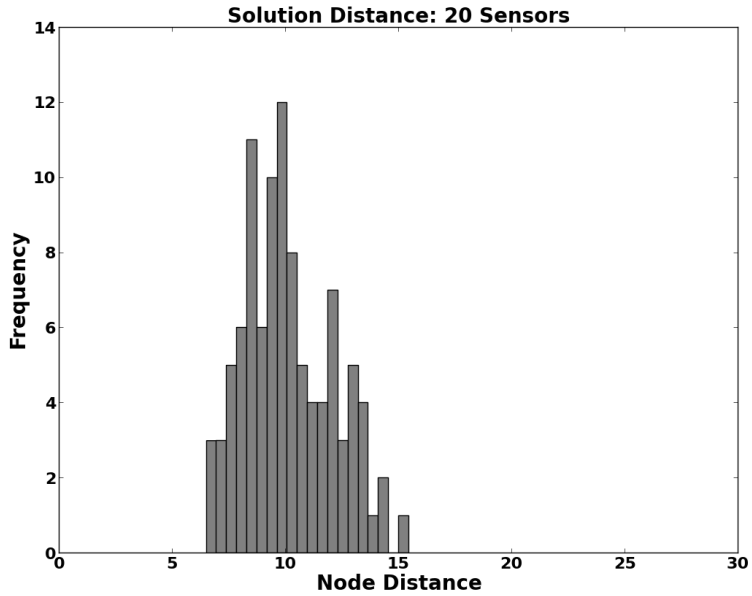

Figure 6.3: Distribution of average number of node hops per sensor to translate candidate solution to trial solutions for 1000 scenarios.

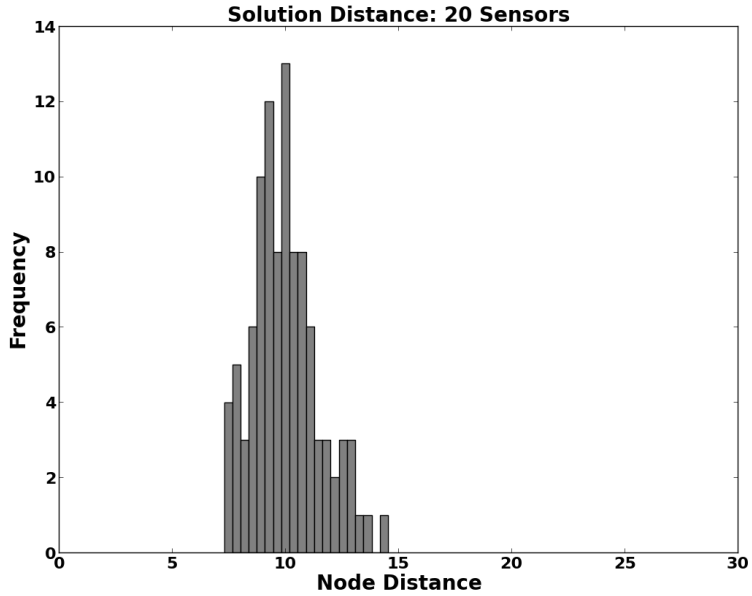

Figure 6.4: Distribution of average number of node hops per sensor to translate candidate solution to trial solutions for 1500 scenarios.

6.3 Summary

In this chapter, a method for analyzing the variability between any two sensor placements was shown. This method uses the assignment problem to translate one placement to the other, and can use a variety of distance metrics.

This solution variability technique was tested on a problem similar to the gas detector placement problem. This problem seeks to place contaminate sensors in municipal water networks to determine when the water quality may be dangerous. The simulation of this problem can be done much more rapidly than the CFD simulations needed for the gas detector problem, therefore many more scenarios were available to test the proposed procedures.

The solution variability method was tested on a larger water network, consisting of $3,358$ potential locations and $332,368$ scenarios. The difference between the candidate sensor placement and the trial sensor placements were calculated in terms of the average number of node hops per sensor necessary to translate one placement to the other. These results show that adding more scenarios beyond 1000 scenarios in the sample used to generate a placement provided little decrease in solution variability. This technique, along with the confidence intervals shown in Chapter 5, help address the problem of determining the appropriate number of scenarios, a major problem in generating scenario sets for gas detector placement.

7. USING TEVA-SPOT FOR GAS DETECTOR PLACEMENT

This following is an example of how the TEVA-SPOT toolkit can be used for applications outside the water network realm (Hart et al., 2008). Gas detection, specifically the detection of combustible and toxic gas release events, is a key component of modern process safety. Combustible gas detection relies upon the detection of a gas before it reaches either its lower explosive or lower flammable limit, LEL and LFL, respectively. These limits refer to the gas concentrations at which a dispersed gas cloud in air will allow a flame front to spread when exposed to an ignition source. Toxic gas detectors are designed to detect and alarm at a concentration related to the impact on human health, typically expressed in parts per million (ppm). Usually, combustible gas detectors and toxic gas detectors are calibrated to alarm when they detect some fixed fraction of the LEL, LFL, or toxic gas limits.

The event scenarios for this example were generated by GexCon using their CFD simulation software FLACS (GexCon, 2011). Three data sets are used. For data set 1, a set of 279 release scenarios were generated for a real process facility. The process geometry itself is proprietary, however, it represents the full process geometry (equipment, piping, etc.) for a medium-scale facility. All scenarios were considered to have equal probability of occuring. Concentrations were provided at 994 potential point sensor locations. Point sensors detect contaminant concentrations in terms of a percentage of the lower flammability limit (LFL). The measure selected for the damage coefficient in this example was the "time to detect" for each release scenario and each potential sensor location. The time to detection was selected since it was the available output for the set of release scenarios. Of course, other measures, like the volume of the flammable cloud at the time of detection, could instead be used.

Data were provided for two sensor detection levels for each sensor type: 10%LFL and 30%LFL for point sensors, 1LFLm and 2LFLm for the line-of-sight sensors. For this optimization problem, we used the lower of the two values for only the point sensor types (10%LFL). It should be noted that 9 of the events were not detected by any of the potential sensors at these concentration settings. These events were removed from the data for all results shown here for simplicity. For data set 2, 314 release scenarios were generated for a separate process module. A total of 768 potential point sensor locations were available, and the damage coefficient was the "time to detect" for a scenario at any location. The gas concentration needed to signal detection is 10%LFL. Finally, for data set 3 a total of 145 release scenarios were used, with a total number of potential point sensor locations of 943. As with the previous two data sets, the 10%LFL concentration was needed for detection.

7.1 Creating Impact File

This raw event data was parsed and organized into an impact file of the following format:

```
<potential-sensor-locations>
<number-of-delays> <delay-time1>
```

and all subsequent lines have the following format:

```
<scenario-index> <node-index> <time-of-detection> <impact-value>
```

For the gas detector placement problem, the `<number-of-delays>` is set to 0. Additionally, because data provided by Gexcon was in terms of "time to detection"

of the gas cloud, the <impact-value> for each scenario and node is the same as the <time-to-detection>. The impact file for the gas detector problem in shown in Appendix G.

7.2 Executing the Sensor Placement

To solve the problem, the following command line is used:

```
sp --network=gasdetector --objective=ec --ub=ns,55 --solver=cplexamp
```

The gasdetector network is specified. This will access the gasdetector_ec.impact file that was generated using the data from Gexcon. The objective is to minimize ec, or the extent of the impact. In this case, this is minimizing the expected detection time of all scenarios. An upper bound on ns, or the number of sensors, is set at 55. The solver used is CPLEX 12.2.

7.2.1 Data Set 1

For Data Set 1, the following output is generated when executing the TEVA-SPOT sensor placement command using --ub=ns,55:

```
read_impact_files:  /home/wst/examples/gasdetector_ec.impact

Number of Nodes                 :  994
Number of Contamination Impacts: 6113

------------------------------------------------------------
Sensor placement id:        11397
Number of sensors:          55
```

```
Total cost:                 0
Sensor node IDs:            24 42 45 59 67 81 83 116 121 139 156 212
214 314 336 362 373 390 402 407 410 411 416 423 460 478 481
483 496 499 509 515 539 552 577 589 593 616 633 640 736 752
761 769 781 852 868 886 906 914 947 956 964 965 974

Impact File:                /home/wst/examples/gasdetector_ec.impact
Number of events:           270
Min impact:                 20.0300
Mean impact:                30.3740
Lower quartile impact:      22.8100
Median impact:              25.9000
Upper quartile impact:      32.8500
Value at Risk (VaR) (  5%): 56.2000
TCE                 (  5%): 66.2600
Max impact:                 86.0000
------------------------------------------------------------

Done with sp
```

These results were directly compared to results shown in Legg et al. (2012b,a). Results from Problem (SP) are shown below. Several metrics were displayed to make comparison with the TEVA sensor placement results easier. The results are shown below:

```
Number of Sensors: 55
Selected Sensors [24, 42, 45, 59, 67, 81, 83, 116, 121, 139, 156, 212,
```

```
214, 314, 336, 362, 373, 390, 402, 407, 410, 411, 416, 423, 460, 478,
481, 483, 496, 499, 509, 515, 539, 552, 577, 589, 593, 616, 633, 640,
736, 752, 761, 769, 781, 852, 868, 886, 906, 914, 947, 956, 964, 965,
974]
Number of events: 270
Minimum detection time: 20.03
Objective (min. expected value):  30.374
Maximum detection time: 86.0
FRACTION DETECTED:  1.0
```

For Data Set 1, all sensors nodes chosen were the same, as well as the important metrics such as `Min impact`, `Mean impact`, and `Max impact`.

7.2.2 Data Set 2

For Data Set 2, the following output is generated when executing the TEVA-SPOT sensor placement command using `--ub=ns,12`:

```
read_impact_files:  /home/wst/examples/gasdetector2_ec.impact

Number of Nodes                :  768
Number of Contamination Impacts: 76800

------------------------------------------------------------
Sensor placement id:        20375
Number of sensors:          12
Total cost:                 0
Sensor node IDs:            53 69 112 128 144 190 214 377 445 627
```

```
629 753

Impact File:                /home/wst/examples/gasdetector2_ec.impact
Number of events:           314
Min impact:                 15.3300
Mean impact:                25.6143
Lower quartile impact:      16.0300
Median impact:              17.9100
Upper quartile impact:      22.1100
Value at Risk (VaR) (  5%): 57.1000
TCE                 (  5%): 116.7756
Max impact:                 518.5000
------------------------------------------------------------

Done with sp
```

Results from Problem (SP) are shown below. Several metrics were displayed to make comparison with the TEVA sensor placement results easier. The results are shown below:

```
Number of Sensors: 12
Selected Sensors [53, 69, 112, 128, 144, 190, 214, 377, 445, 627, 629,
753]

Number of events: 314
Minimum detection time: 15.33
Objective (min. expected value):  25.6142675159
```

```
Maximum detection time: 518.5
FRACTION DETECTED:  1.0
```

All sensors nodes chosen were the same for data set 2. Additionally, the important metrics such as `Min impact`, `Mean impact`, and `Max impact` were also the same as those generated by TEVA-SPOT.

7.2.3 Data Set 3

The following output is generated when executing the TEVA-SPOT sensor placement command using `--ub=ns,12` with data set 3:

```
read_impact_files:  /home/wst/examples/gasdetector3_ec.impact

Number of Nodes                 :  943
Number of Contamination Impacts: 8279

------------------------------------------------------------
Sensor placement id:        20571
Number of sensors:          25
Total cost:                 0
Sensor node IDs:            43 51 95 106 126 140 188 264 279 286 330
364 376 441 591 682 769 813 820 844 870 910 920 932 935

Impact File:                /home/wst/examples/gasdetector3_ec.impact
Number of events:           145
Min impact:                 15.0000
Mean impact:                30.1646
```

```
Lower quartile impact:      18.9000
Median impact:              26.0800
Upper quartile impact:      37.6000
Value at Risk (VaR) (  5%): 62.1000
TCE                 (  5%): 65.0037
Max impact:                 68.2400
------------------------------------------------------------

Done with sp
```

Results from Problem (SP) are shown below. Several metrics were displayed to make comparison with the TEVA sensor placement results easier. The results are shown below:

```
Number of Sensors: 25
Selected Sensors [43, 51, 95, 106, 126, 140, 188, 264, 279, 286, 330,
364, 376, 441, 591, 682, 769, 813, 820, 844, 870, 910, 920, 932,
935]

Number of events: 145
Minimum detection time: 15.0
Objective (min. expected value):  30.1646206897
Maximum detection time: 68.24
FRACTION DETECTED:  1.0
```

For Data Set 3, all sensors nodes chosen were the same, as well as the important metrics such as `Min impact`, `Mean impact`, and `Max impact`.

7.3 Summary

The TEVA-SPOT toolkit was able to duplicate the results of a tailored sensor placement formulation with a little intuitive modification of the impact file to fit the specifications of the gas detector placement problem. Using three independent data sets provided by GexCon, we were able to utilize the TEVA-SPOT sensor placement command to generate sensor placements exactly identical to those placement generated by our tailored sensor placement formulation (SP), as presented in Chapter 2. This toolkit provides an easily accessible tool to users wishing to generate optimal sensor placements without the need of mathematical programming knowledge. By mimicking the impact file format native to TEVA-SPOT, it is easy to use the sensor placement tools that were originally designed for sensor placement in water networks to place sensors in petrochemical facilities. A template for the impact file was shown, as well as the necessary modifications needed to match the needs of the gas detector placement problem.

8. SUMMARY, CONCLUSIONS, AND FUTURE WORK*

In this dissertation, a set of mixed-integer stochastic programming formulations for the optimal placement of gas detectors were presented. These formulations seek to optimally place a finite number of gas detectors given a large set of gas release scenarios representing the uncertainty space of possible releases. The uncertainty in this problem arises from the particular leak locations, leak properties, process characteristics, weather conditions, and process geometries that may be plausible for a given release.

Each problem formulation presented in this document was evaluated using three independent data sets provided by GexCon. Each data set contained contained a set of potential release scenarios generated using their CFD dispersion software, FLACS. These scenarios represented realistic realizations of potential releases on a real, proprietary process geometry and varied in the locations, properties and weather conditions impacting the scenario. For all results in this paper, we selected the objective function to be the expected time to detection across all scenarios. Other metrics, like volume of release at detection, number of impacted indivdiuals, or the expected impact of an explosion could easily be utilized with no change in the problem formulations. Data set 1 was comprised of a set of 270 detectable scenarios with 994 potential point detector locations. Data set 2 had 314 scenarios with 768 potential locations, while data set 3 was made up of 145 detectable scenarios with 943

*Part of this section is reprinted with permission from "A Stochastic Programming Approach For Gas Detector Placement Using CFD-based Dispersion Simulations" by Legg SW, Benavides-Serrano AJ, Siirola JD, Watson JP, Davis SG, Bratteteig A, and Laird CD, 2012. Computers & Chemical Engineering, 47(0):194-201, Copyright 2012 by Elsevier Ltd.

Part of this section is reprinted with permission from "Optimal Gas Detector Placement Under Uncertainty Considering Conditional-Value-at-Risk" by Legg SW, Wang C, Benavides-Serrano AJ, and Laird CD, 2013. Journal of Loss Prevention in the Process Industries, 26(3):410-417, Copyright 2013 by Elsevier Ltd.

potential locations. Each scenario was considered "detected" by a particular sensor location when the concentration of the released gas achieved 10%LFL, and the time at which detection was signalled was recorded as the detection time. All scenarios in each data set were assumed to have an equal probability of occurring, thus α_a is the same for all $a \in A$. While the process geometries themselves are proprietary, they represent the full process geometry (equipment, piping, support structures, etc.) for a facility, and are all approximately 20 m in height and 50 m by 70 m in area.

Each problem formulation was implemented in the Pyomo optimization modeling framework (Hart et al., 2011), and solved using CPLEX 12.2. The formulations provide a provably optimal solution while also being computationally efficient. Most problems required a few seconds on a dual quad-core Intel(R) Xeon(R) CPU X5482 with a clock speed of 3.2GHz and 18 GB RAM for the data sets used. Solution time is similar on a standards dual-core laptop or desktop. Some problem formulations, specifically SP-CVaR, required anywhere from a few minutes to half an hour, but solutions were still achievable in acceptable amounts of time on the machine mentioned above.

8.1 Summary of Problem Formulations and Results

Problem (SP), the first and most general problem formulation, was presented in Chapter 2. An adaptation to a previously developed formulation developed for the placement of water quality sensors in municipal water networks Berry et al. (2004), Problem (SP) sought to minimize the expected detection time across the entire set of release scenarios by optimally locating a finite number of available detectors. Numerical results for this formulation were provided using the data sets mentioned previously. This problem formulation was computationally efficient and also provably optimal with respect to the particular set of scenarios used to generate the extensive

form of the stochastic program. This formulation was shown to work on each of the three data sets. First, the fraction of events detected and the expected detection time was provided for a maximum number of sensors ranging from 1 to 100. This is a useful result for determining the minimum number of detectors needed to detect all of the potential scenarios, as well as determining at which point the addition of a new sensor provides little or no value in terms of risk reduction. For each data set, all scenarios were detected. Additionally, the layout of selected sensors given a particular number of sensors was shown for each data set. These results demonstrate the practicality and usefulness of stochastic programming for the placement of gas detectors, and motivate continued work.

In Chapter 3, an extension, referred to as Problem (SPC), to the original problem formulation (SP) was proposed that included a constraint on the volumetric coverage of sensors throughout the facility. This extension limited the maximum distance between any potential sensor location and the nearest chosen sensor location. In effect, this constraint mandates that all areas of the facility be "covered" by a sensor, regardless of whether there are scenarios in the available scenario set that impact that area. This constraint is a hybridization of current industry sensor placement techniques that rely on providing some level of coverage through the facility with our proposed optimal placement techniques. This constraint provides some reliability of the sensor placement to scenario sets that do not adequately represent the entire uncertainty space for potential release locations and dispersions. Numerical results were again provided for each of the three data sets. Problem (SPC) was shown to suffer from a minimal increase in the expected detection time across when compared to the original problem (SP), even with the constraint on the coverage distance needing to be satisfied. Furthermore, all scenarios were detected by this new formulation. An overhead view of the facility, with potential and selected sensor locations for

both problem formulations (SP) and (SPC), was again provided. It can be seen how each formulation, while providing similar optimal expected detection times for all scenarios, possesses a different set of selected sensors. Problem formulation (SPC) was provably optimal and computationally efficient, requiring only a few seconds to solve.

Problem (SP) was again modified in Chapter 4, this time in include a constraint on the Conditional-Value-at-Risk, or CVaR. Problem (SP) determines a sensor placement with the optimal expected detection time across all scenarios, or in the case where each scenario has an equal probability of occurrence, the optimal mean detection time. While this mean detection time is optimal, this says little about the distribution of the individual detection times for each scenario. An analysis of the distribution of individual detection times was presented for problem (SP) for each data set, revealing that while the mean detection time of all events was quite low, many events required significantly more time to detect than this mean. It is therefore interesting to develop a formulation aimed at improving the detection time of the scenarios requiring the longest times to detect, while also maintaining a satisfactory mean detection time for the entire distribution. In problem (SP-CVaR), a constraint provides an upper bound on the CVaR value for the distribution, while seeking to minimize the expected detection time across all scenarios. This upper bound is determined through solution of problem (CVaR), which minimizes the CVaR value for a particular θ or percentage of the distribution the formulation is concerned with. Numerical results were presented that compared (SP-CVaR) with problem formulation (SP), and also with a general coverage-only placement (C). For each data set, it was shown that while problem (SP) provided a sensor placement with the best performance in minimizing the expected detection time across all scenarios and the bulk of the scenarios were located near the low end of the distribution of detection times,

several scenarios required much more time to detection indicating poorer worst-case performance. In comparison, problem formulation (SP-CVaR) provided significantly better performance on the tail of the distribution of detection times, while only suffering slightly in the expected detection time for the entire distribution. Both formulations (SP) and (SP-CVaR) were compared to the coverage-only placement (C) for data set 1, revealing a major improvement in performance when comparing the optimization-based sensor placement approaches with the coverage-only approach.

In Chapter 5, sensor placements generated by problem formulations (SP) and (SPC) were assessed in how they handled scenarios not known during the optimization. Because the simulation of leak scenarios is the dominant computational cost, requiring anywhere from hours to days to compute, there exists a desire to limit the number of scenarios needed to determine the optimal sensor placement. However, there is no guarantee that a small number of scenarios will adequately cover the uncertainty space of potential gas dispersions. Therefore, a sampling procedure based on the work of Mak et al. (1999) was outlined. This procedure quantifies the effectiveness of a candidate placement on alternate scenario sets not considered during optimization by sampling from a larger pool of scenarios. Problem formulation (SP) generates sensor placements with good performance over the set of scenarios considered during optimization. This formulation is best when the set of scenarios is large enough to sufficiently cover the full uncertainty space and provides sufficiently small confidence intervals on the optimality gap statistic calculated in the Mak, Morton and Wood procedure. If the confidence interval on the optimality gap indicates that there are not enough scenarios to provide the necessary performance guarantees, two options are available. More CFD simulations can be performed, or the original problem formulation can be modified to improve the reliability. Because the cost of CFD simulations can be prohibitively high, especially to generate enough scenarios

to drastically improve the performance, problem (SPC) is proposed. The inclusion of the coverage constraint in problem (SPC) provides some protection against using a scenario set that may not fully approximate the full realm of potential leak locations and dispersion characteristics. Problem (SPC) sacrifices little in the optimal objective value of the scenarios considered, but provides markedly improved detection for the scenarios that were not considered in the original optimization. Additionally, the confidence intervals on the optimality gap are greatly improved in comparison to those generated by problem (SP), indicating better reliability across the full uncertainty space. If the number of scenarios available do not provide the necessary performance guarantees, problem (SPC) is recommended for the placement of gas detectors. Both formulations were compared with the traditional coverage-only placement, both showing a much better performance than (C). The coverage-only performance delivered significantly worse expected detection times and fractions of events detected, even while providing a tighter coverage than the optimization based approaches.

In Chapter 6, a method was proposed to determine the variability in the solution (i.e., variation in placements). This method was tested on a municipal water network sensor placement problem, which is similar to the gas detector placement problem, but possesses simulations that are much quicker to generate. This allows for a much larger scenario set to be generated, and for the network size used here, the entire scenario set is available. The distance between any two proposed sensor placements is determined using the assignment problem to solve for the minimum change needed to translate one sensor placement through another. The metric used to in the objective can vary by problem type, but in this case we propose using the euclidean distance or, in the case of the water network problem, the number of node hops needed. A candidate sensor placement can be generated with a small subsample of the full

scenario space, and then can be compared to a large set of trial placements generated using alternate scenario sets. This allows us to determine if there is significant variation in placements over different scenario sets, but also shows if engineers have flexibility to consider other objectives in the placement of sensors.

The use of an off-the-shelf toolkit for the placement of sensors in municipal water networks, TEVA-SPOT, for the placement of gas detectors in petrochemical facilities was explored in Chapter 7. In order to utilize this readily available tool, modification of the native "impact" files was necessary. These modifications were outlined, explaining how this file can be made using the data available from CFD tools. Additionally, the necessary sensor placement command line options were outlined and tested on the available data sets for the gas release test cases. Results from TEVA-SPOT were compared to those generated by problem (SP), revealing that this toolkit provides identical results without the need for mathematical programming knowledge. An understanding of both the TEVA-SPOT software and the gas dispersion problem was necessary to determine the necessary modifications, though future use of TEVA-SPOT can be performed without this knowledge. A template for the impact file was shown, as well as modifications to the command line options necessary to match the needs of the gas detector placement problem.

8.2 Future Work

In the future, there is a desire to develop extensions to these problem formulations. In real facility settings, sensor failure is a threat that must be considered in safety systems. It is possible to develop a two-stage problem that incorporates the potential of a failed detection by one or more sensors (Berry et al., 2006a, 2009; Berman et al., 2007). Additionally, the potential for false alarms and nuisance trips often motivate safety and plant management teams to implement voting procedures

into safety systems, requiring two or more sensors to detect a leak scenario. This voting policy will be incorporated into a modified sensor placement problem formulation. We have previously shown that the use of a coverage constraint in the problem formulation for minimizing the expected detection time has improved solution reliability (Legg et al., 2012a). It would be interesting to extend this constraint to the CVaR formulation presented in this paper. Finally, larger problem instances may require significantly more powerful computing resources and improved tools to determine the optimal sensor placements. The Progressive Hedging tool available the Pyomo software package may be helpful in solving problems in which it is not efficient or possible to solve the extensive form of the stochastic problem.

REFERENCES

ANSI (2003). *ANSI/ISA-RP12.13.02(IEC 61779-6 Mod) - Recommended Practice for the Installation, Operation, and Maintenance of Combustible Gas Detection Instruments*. ANSI, Washington DC.

API (2001). *API Recommended Practice 14C, Recommended Practice for Analysis, Design, Installation, and Testing of Basic Surface Safety Systems for Offshore Production Platforms, 7th ed.* API, Washington DC.

Artzner, P., Delbaen, F., Eber, J., and Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, 9(3):203–228.

Benavides-Serrano, A., Legg, S., Mannan, M., and Laird, C. (2012). A reliability-p-median formulation for optimization of gas detector layout in process facilities. In *AIChE 2012 Annual Meeting, Pittsburgh*. AICHE.

Berman, O., Krass, D., and Menezes, M. (2007). Facility reliability issues in network p-median problems: strategic centralization and co-location effects. *Operations Research*, 55(2):332–350.

Berry, J., Carr, R., Hart, W., Leung, V., Phillips, C., and Watson, J. (2006a). On the placement of imperfect sensors in municipal water networks. In *Proceedings of the 8th Symposium on Water Distribution Systems Analysis, Cincinnati*. ASCE.

Berry, J., Carr, R., Hart, W., Leung, V., Phillips, C., and Watson, J. (2009). Designing contamination warning systems for municipal water networks using imperfect sensors. *Journal of Water Resources Planning and Management*, 135(4):253–263.

Berry, J., Fleischer, L., Hart, W., Phillips, C., and Watson, J. (2005). Sensor placement in municipal water networks. *Journal of Water Resources Planning and Management*, 131(3):237–243.

Berry, J., Hart, W., Phillips, C., and Uber, J. (2004). A general integer-programming-based framework for sensor placement in municipal water networks. In *Critical Transitions In Water And Environmental Resources Management, Salt Lake City*, pages 1–10. ASCE.

Berry, J., Hart, W., Phillips, C., Uber, J., and Watson, J. (2006b). Sensor placement in municipal water networks with temporal integer programming models. *Journal of Water Resources Planning and Management*, 132:218–224.

Bratteteig, A., Hansen, O., Gavelli, F., and Davis, S. (2011). Using cfd to analyze gas detector placement in process facilities. In *2011 Mary Kay O'Connor Process Safety Center International Symposium, College Station*.

Carr, R., Greenberg, H., Hart, W., Konjevod, G., Lauer, E., Lin, H., Morrison, T., and Phillips, C. (2006). Robust optimization of contaminant sensor placement for community water systems. *Mathematical Programming*, 107(1):337–356.

CCPS (2009). *Continuous Monitoring for Hazardous Material Releases*. Wiley-AIChE, New York.

Chou, J. (2000). *Hazardous gas monitors: a practical guide to selection, operation and applications*. McGraw-Hill Professional, New York.

COOPR (2009). Coopr: A common optimization python repository. http://software.sandia.gov/coopr. [Online; accessed 17-December-2011].

DeFriend, S., Dejmek, M., Porter, L., Deshotels, B., and Natvig, B. (2008). A risk-based approach to flammable gas detector spacing. *Journal of hazardous materials*, 159(1):142–151.

Delphian Detection Technology (2013). Combustible gas monitors. http://www.delphian.com/chc.htm. [Online; accessed 8-February-2013].

Det-Tronics (2013). Open path (opecl) ir gas detector. http://www.detronics.com/. [Online; accessed 8-February-2013].

Dhillon, S. and Chakrabarty, K. (2003). *Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks*, volume 3. IEEE, New York.

Duffie, D. and Pan, J. (1997). An overview of value at risk. *The Journal of Derivatives*, 4(3):7–49.

Fire & Safety World Online (2011). Gas detection. http://www.fs-business.com/FAQ/FAQGasDetection.asp. [Online; accessed 19-July-2011].

Gencer, C., Aydogan, E., and Soydemır, A. (2008). Chemical agent detector placement methodology. *Applied Mathematics and Computation*, 195(2):542–557.

GexCon (2010). *FLACS v9.1 User's Manual*. GexCon AS, Washington DC.

GexCon (2011). Flacs cfd disperson modeling and explosion software. http://gexcon.com/FLACSoverview. [Online; accessed 15-January-2012].

Hakimi, S. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, 13(3):462–475.

Hamel, D., Chwastek, M., Farouk, B., Dandekar, K., and Kam, M. (2006). A computational fluid dynamics approach for optimization of a sensor network. In *Measurement Systems for Homeland Security, Contraband Detection and Personal Safety, Proceedings of the 2006 IEEE International Workshop, Boston*, pages 38–42. IEEE.

Hamming, R. (1950). Error-detecting and error-correcting codes. *Bell System Technical Journal*, 29(2):147–160.

Hart, W., Berry, J., Boman, E., Murray, R., Phillips, C., Riesen, L., and Watson, J. (2008). The teva-spot toolkit for drinking water contaminant warning system design. In *Proceedings World Environmental and Water Resources Congress, Reston*. ASCE.

Hart, W., Watson, J., and Woodruff, D. (2011). Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260.

Hjertager, B. (1984). Computer simulation of turbulent reactive gas dynamics. *Modeling, Identification and Control*, 5(4):211–236.

Hjertager, B. (1986). Three-dimensional modeling of flow, heat transfer, and combustion. *Handbook of Heat and Mass Transfer, Gulf Publishing Company, Houston, TX*, pages 304–350.

HSE (1993). *Offshore Detector Siting Criterion Investigation of Detector Spacing by LLoyds Register, in Offshore Technology Report OTO 93002*. HSE, London.

HSL (2001). *Framework for HSE Guidance on Gas Detectors (On-line Checking of Flammability Monitoring Equipment - Final Report)*. HSL, Sheffield.

IBM (2010). *CPLEX 12.2 User's Manual*. IBM, Sunnyvale.

IEC (2007). *IEC 60079, Explosive Atmospheres - Part 29-2: Gas Detectors - Selection, Installation, Use and Maintenance of Detectors for Flammable Gases and Oxygen*. IEC, Geneva.

ISA (2010). *ISA-TR84.00.07-2010: Technical Report Guidance on the Evaluation of Fire, Combustible Gas and Toxic Gas System Effectiveness*. ISA, Washington DC.

Kelsey, A., Hemingway, M., Walsh, P., and Connolly, S. (2002). Evaluation of flammable gas detector networks based on experimental simulations of offshore, high pressure gas releases. *Process Safety and Environmental Protection*, 80(2):78–86.

Kelsey, A., Ivings, M., Hemingway, M., Walsh, P., and Connolly, S. (2005). Sensitivity studies of offshore gas detector networks based on experimental simulations of high pressure gas releases. *Process Safety and Environmental Protection*, 83(3):262–269.

Kessler, A., Ostfeld, A., and Sinai, G. (1998). Detecting accidental contaminations in municipal water networks. *Journal of Water Resources Planning and Management*, 124(4):192–198.

Krokhmal, P., Zabarankin, M., and Uryasev, S. (2011). Modeling and optimization of risk. *Surveys in Operations Research and Management Science*, 16(2):49–66.

Lee, R. and Kulesz, J. (2008). A risk-based sensor placement methodology. *Journal of hazardous materials*, 158(2):417–429.

Legg, S., Benavides-Serrano, A., Siirola, J., Watson, J., Davis, S., Bratteteig, A., and Laird, C. (2012a). A stochastic programming approach for gas detector place-

ment using cfd-based dispersion simulations. *Computers & Chemical Engineering*, 47(0):194 – 201.

Legg, S., Siirola, J., Watson, J., Davis, S., Bratteteig, A., and Laird, C. (2012b). A stochastic programming approach for gas detector placement in process facilities. In *Proceedings of the FOCAPO Conference, Savannah*. FOCAPO.

Legg, S., Wang, C., Benavides-Serrano, A., and Laird, C. (2013). Optimal gas detector placement under uncertainty considering conditional-value-at-risk. *Journal of Loss Prevention in the Process Industries*, 26(3):410 – 417.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710.

Mak, W., Morton, D., and Wood, R. (1999). Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24(1):47–56.

Marx, J. and Cornwell, J. (2009). The importance of weather variations in a quantitative risk analysis. *Journal of Loss Prevention in the Process Industries*, 22(6):803–808.

Mirchandani, P. and Francis, R. (1990). *Discrete Location Theory*, volume 23. Wiley-Interscience, New Jersey.

NFPA (2007). *NFPA 15, Standard for Water Spray Fixed Systems for Fire Protection*. National Fire Protection Association, Quincy.

Nolan, D. (1996). *Handbook of Fire and Explosion Protection Engineering Principles: For Oil, Gas, Chemical and Related Facilities*. William Andrew Publishing, New York.

Nolan, D. (2010). *Handbook of fire and explosion protection engineering principles: for oil, gas, chemical and related facilities*. William Andrew Publishing, New York.

Obenschain, K., Boris, J., and Patnaik, G. (2004). Using ct-analyst to optimize sensor placement. In *Defense and Security, Kissimmee*, volume 5416, pages 14–20.

Ostfeld, A. and Salomons, E. (2004). Optimal layout of early warning detection stations for water distribution systems security. *Journal of Water Resources Planning and Management*, 130(5):377–385.

Resende, M. and Werneck, R. (2004). A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10(1):59–88.

ReVelle, C. and Swain, R. (1970). Central facilities location. *Geographical Analysis*, 2(1):30–42.

Rockafellar, R. and Uryasev, S. (2000). Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42.

Rockafellar, R. and Uryasev, S. (2002). Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 26(7):1443–1471.

Shastri, Y. and Diwekar, U. (2006). Sensor placement in water networks: A stochastic programming approach. *Journal of Water Resources Planning and Management*, 132(3):192–203.

Strøm, Ø. and Bakke, J. (1999). Gas detector location. *Safety on offshore installations*, pages 3.3.1–3.3.12.

UK Health & Safety Executive (2011a). Fire and gas detection. http://www.hse.gov.uk/offshore/strategy/fgdetect.htm. [Online; accessed 19-July-2011].

UK Health & Safety Executive (2011b). Selection and use of flammable gas detectors. http://www.hse.gov.uk/pubns/gasdetector.pdf. [Online; accessed 6-February-2013].

UKOOA (1995). *Guidelines for Fire and Explosion Hazard Management*. UKOOA, London.

Uryasev, S. (2000). Conditional value at risk: Optimization algorithms and applications. *Financial Engineering News*, 14:1–5.

Watson, J., Greenberg, H., and Hart, W. (2004). A multi-objective analysis of sensor placement optimization in water networks. In *Proceedings of the ASCE/EWRI Conference, Salt Lake City*.

Watson, J., Murray, R., and Hart, W. (2009). Formulation and optimization of robust sensor placement problems for drinking water contamination warning systems. *Journal of Infrastructure Systems*, 15(4):330–339.

Yujian, L. and Bo, L. (2007). A normalized levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095.

APPENDIX A

NOTATION

The following notation is used for the MILP formulations in this text:

L	Set of N candidate detector locations
N	Number of candidate sensor locations
l	Sensor location index
A	Set of M leak scenarios
M	Number of release events
a	Release event index
$\mathcal{L}_a$	Candidate sensor locations affected by leak scenario a
α_a	Probability of leak scenario a
$d_{a,i}$	Damage coefficient for leak scenario a at location i
p	Maximum number of sensors allowed
s_l	Binary variable indicating if a sensor is installed at location l
$x_{a,i}$	Indicator for location i that first detects scenario a
q	Dummy sensor location for undetected events
t_{max}	Maximum time to detect
C_l	Set of sensor locations that provide coverage to location l
c	Coverage index

APPENDIX B

EXAMPLE SENSOR PLACEMENT RUN FILE

This is the `sensor_placement.py` file. This file will build the extensive forms of the stochastic sensor placement problems, call the necessary solver, and return the results. The model file is named `sensor_placement_model.py` and is shown in Appendix C.

```
import sys, random, gc
if len(sys.argv) != 7:
    print "Command line should read: '~/$ python sensor_placement.py\
     <seed> <num scenarios> <max sensors> <min_(exp or undet or \
     max) or CVaR> <cvar val> <cov level>' "
    sys.exit()


from coopr.pyomo import *
from pyutilib.misc import Options
from sensor_placement_model import *
from parsed_sensor_locations import *
from coopr.opt import SolverFactory

# import the gexcon data
sys.path.append('data_files/data')
from data import *

solver = SolverFactory('cplex',solver_io='lp')
```

```
print_output = False

seed_number = sys.argv[1]
num_scenarios = sys.argv[2]
max_sensors = int(sys.argv[3])
obj_type_status = sys.argv[4]
cvar_star = float(sys.argv[5])
coverage_status = sys.argv[6]

# create the set of random scenarios
valid_scenarios = set()
for scen in detected_scenarios:
    if len(detection_dict[scen]) > 0:
        valid_scenarios.add(scen)

if len(valid_scenarios) < int(num_scenarios):
    num_scenarios = len(valid_scenarios)

if (seed_number == 'clock'):
    random.seed(None)
else:
    random.seed(int(seed_number))

coverage_dict = None
if coverage_status == 'none':
    coverages = dict()
```

```
else:
    cvgmod = __import__(str(int(coverage_status))+'m_cvg')
    coverages = cvgmod.coverage_dict

random_scenarios = sorted(random.sample(valid_scenarios, \
int(num_scenarios)))
#print "Currently using the following scenarios:"
#print random_scenarios

max_damage = 510.0
# Max Damage is 510 for old data, 900 for data2, and 510 for data3
# create the model
model = create_sensor_placement_model(random_scenarios, \
useful_locations, detection_dict, damage_dict, max_sensors,\
obj_type_status, cvar_star, coverages)
instance = model.create()

#(results, junk) = scripting.util.apply_optimizer(opt, instance)
results = solver.solve(instance,tee=print_output)
stat = str(results.Solver.Termination_condition)

if stat == 'optimal':
    instance.load(results)
#    display(instance.x)
    objective = (value(instance.obj))
```

```
    scen_damage = dict()
    for a in random_scenarios:
        scen_damage[a] = max_damage
        for l in detection_dict[a]:
            if value(instance.s[l]) > 0.001:
                if damage_dict[a][l] < scen_damage[a]:
                    scen_damage[a] = damage_dict[a][l]

    obj_recalc = 0

#    display(instance.s)
    sensors_used = []
    for sens in useful_locations:
        if value(instance.s[sens]) > 0.01:
            sensors_used.append(sens)

    for a in random_scenarios:
        obj_recalc = obj_recalc + scen_damage[a]
    obj_recalc = float(obj_recalc)*(1.0/float(len(random_scenarios)))

    print objective, obj_recalc, 0
    for a in random_scenarios:
        print str(scen_damage[a]) + ' ',
    print

#    print x coordinates
```

```
    for k in sensors_used:
        v=sens_coord[int(k)]
        print v[0],
    print

#    print y coordinates
    for k in sensors_used:
        v=sens_coord[int(k)]
        print v[1],
    print



elif stat == 'infeasible':
    objective = -1
    num_sens = -1
    fraction_detected = -1
    print 'Problem infeasible.'
else:
    print 'Bad things are happening. Trust nothing.'
    sys.exit()
```

APPENDIX C

EXAMPLE PYOMO MODEL FILES

This is the Pyomo file containing all of the mixed-integer sensor placement formulations described in this document. This file is named `sensor_placement_model.py`.

```
from coopr.pyomo import *

def create_sensor_placement_model(scenarios_to_use, sensors_to_use,\
sensor_impact_dict, damage_coeffs, max_sensors, obj_choice,\
cvar_star, coverages):
    # sensor_impact_dict is a dictionary indexed by scenarios that \
     #returns a list of all sensor locations that impact that \
     #scenario
    # damage_coeffs is a python dictionary that maps a tuple of \
     #(scenario_idx, sensor_idx) to the damage coefficient



    model = ConcreteModel()


    full_sensor_impact_tuples = [(scenario, sensor) for scenario in \
     scenarios_to_use for sensor in sensor_impact_dict[scenario]]
    sensor_impact_tuples = list()
    for (scenario,sensor) in full_sensor_impact_tuples:
        if (scenario in scenarios_to_use and sensor in \
         sensors_to_use):
```

```
        sensor_impact_tuples.append( (scenario,sensor) )


model.x = Var(sensor_impact_tuples, bounds=(0.0,1.0))
model.s = Var(sensors_to_use, within=Binary)

if obj_choice == 'min_exp':
    model = create_min_exp_model(model, scenarios_to_use, \
     sensors_to_use, sensor_impact_dict, damage_coeffs, \
     max_sensors, sensor_impact_tuples)
elif obj_choice == 'cvar':
    model = create_cvar_model(model, scenarios_to_use, \
     sensors_to_use, sensor_impact_dict, damage_coeffs,\
     max_sensors, sensor_impact_tuples)
elif obj_choice == 'min_expcvar':
    model = create_min_expcvar_model(model, scenarios_to_use,\
     sensors_to_use, sensor_impact_dict, damage_coeffs, \
     max_sensors, sensor_impact_tuples, cvar_star)
elif obj_choice == 'min_max':
    model = create_min_max_model(model, scenarios_to_use, \
     sensors_to_use, sensor_impact_dict, damage_coeffs, \
     max_sensors, sensor_impact_tuples)
elif obj_choice == 'min_expcov':
    model = create_min_expcov_model(model, scenarios_to_use,\
     sensors_to_use, sensor_impact_dict, damage_coeffs, \
     max_sensors, sensor_impact_tuples, coverages)
```

```
    return model


def create_min_exp_model(model, scenarios_to_use, sensors_to_use,\
sensor_impact_dict, damage_coeffs, max_sensors, \
sensor_impact_tuples):

    def obj_rule (model):
        return 1.0/len(scenarios_to_use)*(sum(damage_coeffs[scen]\
         [sens]*model.x[(scen, sens)] for (scen,sens) in \
         sensor_impact_tuples))
    model.obj = Objective(rule = obj_rule)


    def max_sensors_rule(model):
        return sum(model.s[sens] for sens in sensors_to_use) <= \
         max_sensors
    model.max_sensors_con = Constraint(rule = max_sensors_rule)

    def sensor_binary_rule(model, scen, sens):
        return model.x[(scen, sens)] <= model.s[sens]
    model.sensor_binary_con = Constraint(sensor_impact_tuples, \
     rule = sensor_binary_rule)

    def event_detection_rule(model, scen):
```

```
        return sum(model.x[(scen,sens)] for sens in \
         sensor_impact_dict[scen]) == 1
    model.event_detection_con = Constraint(scenarios_to_use, rule \
     = event_detection_rule)

    return model

def create_min_expcov_model(model, scenarios_to_use, sensors_to_use,\
sensor_impact_dict, damage_coeffs, max_sensors, \
sensor_impact_tuples, coverages):

    def obj_rule (model):
        return 1.0/len(scenarios_to_use)*(sum(damage_coeffs[scen]\
         [sens]*model.x[(scen, sens)] for (scen,sens) in \
         sensor_impact_tuples))
    model.obj = Objective(rule = obj_rule)


    def max_sensors_rule(model):
        return sum(model.s[sens] for sens in sensors_to_use) <= \
         max_sensors
    model.max_sensors_con = Constraint(rule = max_sensors_rule)

    def sensor_binary_rule(model, scen, sens):
        return model.x[(scen, sens)] <= model.s[sens]
    model.sensor_binary_con = Constraint(sensor_impact_tuples, \
```

```
 rule = sensor_binary_rule)

def event_detection_rule(model, scen):
    return sum(model.x[(scen,sens)] for sens in \
     sensor_impact_dict[scen]) == 1
model.event_detection_con = Constraint(scenarios_to_use, \
 rule = event_detection_rule)

def coverage_rule(model, sens):
    expr = 0
    cvgcntr = 0
    at_least_one_set = False
    for j in coverages[sens]:
        if (j in sensors_to_use):
            expr += model.s[j]
            at_least_one_set = True

    if at_least_one_set:
        return expr >= 1
    return Constraint.Skip
if (len(coverages) > 0):
    model.coverage_con = Constraint(sensors_to_use, rule = \
     coverage_rule)


return model
```

```
def create_cvar_model(model, scenarios_to_use, sensors_to_use, \
sensor_impact_dict, damage_coeffs, max_sensors, \
sensor_impact_tuples):

    model.z = Var(scenarios_to_use, bounds=(0.0,None))
    model.beta = Var()
    model.alphaCVaR = Param(default=0.85)



    def obj_cvar (model):
        return model.beta + (1.0/(1.0-model.alphaCVaR)) * \
         (1.0/len(scenarios_to_use)) * (sum(model.z[scen] for scen \
         in scenarios_to_use))
    model.obj = Objective(rule = obj_cvar)



    def cvar_con_rule (model, scen):
        return sum(damage_coeffs[scen][sens]*model.x[(scen, sens)] \
         for sens in sensor_impact_dict[scen]) - model.beta <= \
         model.z[scen]
    model.cvar_con = Constraint(scenarios_to_use, rule = \
     cvar_con_rule)
```

```
    def max_sensors_rule(model):
        return sum(model.s[sens] for sens in sensors_to_use) <= \
         max_sensors
    model.max_sensors_con = Constraint(rule = max_sensors_rule)

    def sensor_binary_rule(model, scen, sens):
        return model.x[(scen, sens)] <= model.s[sens]
    model.sensor_binary_con = Constraint(sensor_impact_tuples, \
     rule = sensor_binary_rule)

    def event_detection_rule(model, scen):
        return sum(model.x[(scen,sens)] for sens in \
         sensor_impact_dict[scen]) == 1
    model.event_detection_con = Constraint(scenarios_to_use, \
     rule = event_detection_rule)

    return model

def create_min_expcvar_model(model, scenarios_to_use, \
sensors_to_use, sensor_impact_dict, damage_coeffs, \
max_sensors, sensor_impact_tuples, cvar_star):

    model.z = Var(scenarios_to_use, bounds=(0.0,None))
    model.beta = Var()
    model.alphaCVaR = Param(default=0.85)
```

```
def obj_rule (model):
    return 1.0/len(scenarios_to_use)*(sum(damage_coeffs\
     [scen][sens]*model.x[(scen, sens)] for (scen,sens) \
     in sensor_impact_tuples))
model.obj = Objective(rule = obj_rule)



def cvar_star_con (model):
    return model.beta + (1.0/(1.0-model.alphaCVaR)) * \
     (1.0/len(scenarios_to_use)) * (sum(model.z[scen] \
     for scen in scenarios_to_use)) <= cvar_star
model.cvar_star_con = Constraint(rule = cvar_star_con)

def cvar_con_rule (model, scen):
    return sum(damage_coeffs[scen][sens]*model.x[(scen, sens)] \
     for sens in sensor_impact_dict[scen]) - model.beta <= \
     model.z[scen]
model.cvar_con = Constraint(scenarios_to_use, rule = \
 cvar_con_rule)

def max_sensors_rule(model):
    return sum(model.s[sens] for sens in sensors_to_use) <= \
     max_sensors
model.max_sensors_con = Constraint(rule = max_sensors_rule)

def sensor_binary_rule(model, scen, sens):
```

```
        return model.x[(scen, sens)] <= model.s[sens]
    model.sensor_binary_con = Constraint(sensor_impact_tuples, \
     rule = sensor_binary_rule)

    def event_detection_rule(model, scen):
        return sum(model.x[(scen,sens)] for sens in \
         sensor_impact_dict[scen]) == 1
    model.event_detection_con = Constraint(scenarios_to_use, \
     rule = event_detection_rule)

    return model


def create_min_max_model(model, scenarios_to_use, \
sensors_to_use, sensor_impact_dict, damage_coeffs, \
max_sensors, sensor_impact_tuples):

    model.max_time = Var()

    def obj_rule (model):
        return model.max_time
    model.obj = Objective(rule = obj_rule)


    def calc_max_time_detect(model,scen):
        return sum(damage_coeffs[scen][sens]*model.x[(scen, \
```

```
     sens)] for sens in sensor_impact_dict[scen]) <= \
     model.max_time
model.calc_max_time_detect = Constraint(scenarios_to_use,\
 rule = calc_max_time_detect)

def max_sensors_rule(model):
    return sum(model.s[sens] for sens in sensors_to_use) <=\
     max_sensors
model.max_sensors_con = Constraint(rule = max_sensors_rule)

def sensor_binary_rule(model, scen, sens):
    return model.x[(scen, sens)] <= model.s[sens]
model.sensor_binary_con = Constraint(sensor_impact_tuples, \
 rule = sensor_binary_rule)

def event_detection_rule(model, scen):
    return sum(model.x[(scen,sens)] for sens in \
     sensor_impact_dict[scen]) == 1
model.event_detection_con = Constraint(scenarios_to_use, \
 rule = event_detection_rule)

return model
```

APPENDIX D

EXAMPLE DATA FILE

This is an appended example of the data file `data.py`, which is used to construct the extensive forms of the stochastic programming formulations by the run file `sensor_placement.py`.

```
all_locations = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10',\
'11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21',\
'22', '23', '24', '25', '26', '27', '28', '29', '30', '31', '32',\
'33', '34', '35', '36', '37', '38', '39', '40', '41', '42', '43',\
'44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54',\
'55', '56', '57', '58', '59', '60', '61', '62', '63', '64', '65',\
'66', '67', '68', '69', '70', '71', '72', '73', '74', '75', '76',\
'77', '78', '79', '80', '81', '82', '83', '84', '85', '86', '87',\
'88', '89', '90', '91', '92', '93', '94', '95', '96', '97', '98',\
'99']


useful_locations = set(['56', '77', '54', '76', '59', '60', \
'61', '62', '63', '64', '65', '67', '82', '83', '81', '86', '87', \
'84', '85', '24', '25', '26', '27', '20', '21', '22', '23', '58', \
'47', '28', '29', '94', '1', '2', '5', '4', '7', '6', '9', '8', \
'96', '68', '97', '99', '98', '75', '74', '73', '72', '71', '70', \
'91', '90', '93', '92', '69', '95', '79', '78', '11', '10', '13', \
'12', '15', '14', '17', '16', '33', '32', '57', '30', '51', '50', \
'53', '52', '19', '55', '89', '88', '18', '31'])
```

```
all_scenarios = ['111310', '111330', '112310', \
'112330', '113310', '113330', '114310', '114330', \
'115310', '115330', '116310', '116330', '117310', \
'117330', '121310', '121330', '122310', '122330', \
'123310', '123330', '125310', '125330', '126310', \
'126330', '127310', '127330', '132310', '132330', \
'134310', '134330', '135310', '135330', '137310', \
'137330', '141310', '141330', '143310', '143330', \
'144310', '144330', '146310', '146330', '147310', \
'147330', '152310', '152330', '153310', '153330', \
'154310', '154330', '156310', '156330', '157310', \
'157330', '211310', '211330', '212310', '212330', \
'213310', '213330', '214310', '214330', '215310', \
'215330', '217310', '217330', '221310', '221330', \
'222310', '222330', '223310', '223330', '224310', \
'224330', '225310', '225330', '227310', '227330', \
'233310', '233330', '235310', '235330', '237310', \
'237330', '253310', '253330', '255310', '255330', \
'257310', '257330', '261310', '261330', '262310', \
'262330', '263310', '263330', '264310', '264330', \
'265310', '265330', '266310']

detected_scenarios = set(['121330', '264330', '154330',\
 '223310', '113310', '121310', '152310', '134330', \
 '125310', '141310', '257330', '227310', '215310', \
```

```
 '223330', '146330', '112310', '157330', '253310', \
 '117310', '222310', '211330', '221330', '112330', \
 '222330', '117330', '237330', '115330', '126330', \
 '237310', '227330', '157310', '253330', '115310', \
 '214330', '137330', '262310', '221310', '126310', \
 '132310', '264310', '137310', '156310', '153310', \
 '116330', '156330', '116310', '153330', '266310', \
 '127310', '233330', '261310', '225330', '257310', \
 '127330', '211310', '214310', '224330', '263330', \
 '217330', '113330', '212330', '224310', '265330', \
 '233310', '263310', '114310', '225310', '212310'])

detection_dict = {'154310': set([]), '215310': set(['11', '32']), \
'141310': set(['32']), '123330': set([]), '261310': \
set(['25', '32']), '144330': set([]), '157330': set(['25', '32']), \
'235310': set([]), '134330': set(['11']), '253330': \
set(['15', '17', '16']), '115310': set(['11', '33', '21']), \
'137330': set(['13']), '143330': set([]), '126310': \
set(['60', '61', '62', '67', '68', '69', '84', '85', '24', '23', \
'7', '77', '76', '75', '74', '70', '93', '94', '59', '58', '55', \
'54', '31', '51', '50', '53', '52']), '132310': set(['12']), \
'264310': set(['5']), '111310': set([]), '214310': set(\
['21', '22', '23', '19', '33', '9']), '116310': set(['11', \
'10', '22', '33', '31', '29', '24']), '127310': \
set(['2', '5', '4']), '257330': set(['99', '15', '14', '17', \
'16', '18', '30', '98', '1', '91', '90', '82', '83', '92']), \
```

```
’122330’: set([]), ’224330’: set([’60’, ’15’, ’51’, \
’52’, ’1’, ’76’, ’75’, ’2’, ’5’, ’68’, ’59’, ’84’]), ’135330’:\
set([]), ’265330’: set([’32’]), ’225310’: \
set([’14’, ’23’]), ’121330’: set([’4’]), ’215330’: set([]),
’152310’: set([’16’]), ’261330’: set([]), ’141330’: \
set([]), ’223330’: set([’11’, ’10’, ’13’, ’12’, ’28’, ’29’]),\
’146330’: set([’11’, ’18’]), ’253310’: set([’18’, ’15’, \
’16’]), ’117310’: set([’33’, ’20’, ’21’, ’22’]), ’147330’:\
set([]), ’237310’: set([’55’, ’63’, ’7’]), ’143310’: set([]),\
’262310’: set([’16’]), ’126330’: set([’60’, ’61’, ’62’, \
’63’, ’64’, ’67’, ’68’, ’69’, ’86’, ’87’, ’84’, ’85’, ’26’, ’6’, \
’77’, ’76’, ’72’, ’71’, ’70’, ’95’, ’94’, ’79’, ’78’, ’59’, \
’56’, ’51’, ’52’]), ’137310’: set([’13’]), ’211310’: \
set([’9’, ’19’]), ’153310’: set([’9’, ’32’]), ’116330’: \
set([’11’, ’10’, ’29’, ’22’, ’33’]), ’214330’: set([’33’, \
’20’, ’21’, ’22’]), ’265310’: set([]), ’225330’: set([\
’23’, ’14’, ’21’]), ’257310’: set([’82’, ’83’, ’15’, \
’91’, ’16’]), ’127330’: set([’2’]), ’263330’: set([’23’]), \
’233310’:set([’25’, ’26’, ’32’, ’71’, ’6’, ’95’]), ’146310’:\
set([]), ’213330’: set([]), ’223310’: set([’11’, ’10’, ’13’,\
’12’, ’28’, ’29’]), ’152330’: set([]), ’113310’: set([’8’]), \
’121310’: set([’4’]), ’262330’: set([]), ’125330’: set([]),\
’112310’: set([’11’, ’18’, ’21’, ’29’]), ’264330’: \}


damage_dict = {’154310’: {}, ’215310’: {’11’: 121.9, \
’32’: 133.5}, ’141310’: {’32’: 47.6}, ’123330’: {}, \
```

```
’261310’: {’25’: 217.7, ’32’: 46.8}, ’144330’: {}, \
’157330’: {’25’: 67.78, ’32’: 71.08}, ’235310’: {}, \
’134330’: {’11’: 43.7}, ’253330’: {’15’: 42.3, ’17’: \
101.4, ’16’: 53.8}, ’115310’: {’11’: 350.5, ’33’: 48.0,\
’21’: 39.0}, ’137330’: {’13’: 20.86}, ’143330’: {}, \
’126310’: {’60’: 19.3, ’61’: 22.6, ’62’: 26.4, ’67’: 64.2,\
’68’: 23.6, ’69’: 32.6, ’84’: 52.5, ’85’: 48.8, ’24’: \
101.8, ’23’: 294.6, ’7’: 55.7, ’77’: 381.3, ’76’: 38.7, \
’75’: 176.1, ’74’: 459.9, ’70’: 39.6, ’93’: 69.6, ’94’: \
80.3, ’59’: 95.3, ’58’: 168.3, ’55’: 51.7, ’54’: 30.6, ’31’: \
164.5, ’51’: 22.2, ’50’: 35.9, ’53’: 19.9, ’52’: 16.8}, \
’132310’: {’12’: 25.4}, ’264310’: {’5’: 19.1}, ’111310’: \
{}, ’214310’: {’21’: 67.6, ’22’: 18.5, ’23’: 64.7, ’19’: \
332.3, ’33’: 23.4, ’9’: 280.9}, ’116310’: {’11’: 149.6, \
’10’: 406.8, ’22’: 50.2, ’33’: 49.5, ’31’: 117.7, ’29’: \
57.6, ’24’: 94.9}, ’127310’: {’2’: 33.01, ’5’: 82.21, ’4’: \
30.4}, ’257330’: {’99’: 38.01, ’15’: 43.69, ’14’: 47.44, \
’17’: 155.3, ’16’: 77.67, ’18’: 156.0, ’30’: 83.69, ’98’: \
111.4, ’1’: 32.79, ’91’: 24.26, ’90’: 80.17, ’82’: 18.24, \
’83’: 21.08, ’92’: 34.49}, ’122330’: {}, ’224330’: {’60’: \
46.2, ’15’: 105.4, ’51’: 74.6, ’52’: 67.7, ’1’: 21.3, ’76’: \
27.1, ’75’: 37.0, ’2’: 98.7, ’5’: 111.1, ’68’: 34.8, ’59’: \
63.2, ’84’: 29.0}, ’135330’: {}, ’265330’: {’32’: 61.2}, \
’225310’: {’14’: 34.7, ’23’: 66.1}, ’121330’: {’4’: 21.3}, \
’215330’: {}, ’152310’: {’16’: 62.3}, ’261330’: {}, \
’141330’: {}, ’223330’: {’11’: 78.6, ’10’: 69.8, ’13’: 42.2,\
```

```
’12’: 80.2, ’28’: 93.6, ’29’: 31.5}, ’146330’: {’11’: 81.3, \
’18’: 47.0}, ’253310’: {’18’: 102.3, ’15’: 270.3, ’16’: \
142.2}, ’117310’: {’33’: 36.88, ’20’: 177.2, ’21’: 32.9, \
’22’: 29.26}, ’147330’: {}, ’237310’: {’55’: 20.4, ’63’: \
56.08, ’7’: 31.54}, ’143310’: {}, ’262310’: {’16’: 51.4}, \
’126330’: {’60’: 19.3, ’61’: 20.7, ’62’: 23.5, ’63’: 33.6, \
’64’: 58.6, ’67’: 107.8, ’68’: 23.9, ’69’: 29.5, ’86’: \
136.6, ’87’: 320.6, ’84’: 89.9, ’85’: 50.5, ’26’: 103.3, \
’6’: 46.8, ’77’: 101.6, ’76’: 34.4, ’72’: 60.5, ’71’: 34.2, \
’70’: 27.5, ’95’: 229.5, ’94’: 110.1, ’79’: 279.1, ’78’: \
109.9, ’59’: 141.5, ’56’: 212.7, ’51’: 21.3, ’52’: 17.0}, \
’137310’: {’13’: 21.99}, ’211310’: {’9’: 145.1, ’19’: 52.0}}
```

APPENDIX E

EXAMPLE COVERAGE DICTIONARY FILE

This is an appended coverage dictionary file for `sensor_placement.py`. This file allows for the coverage option to be implement. This includes the sets of locations that cover each other to a 2 m coverage distance. This file is named `2m_cvg.py`.

```
coverage_dict = {'344': set(['344', '155', '443', '245', \
'533', '438', '439', '250', '249', '434', '254']), '345': \
set(['10', '435', '156', '534', '440', '444', '345', '246', \
'439', '250', '251', '255']), '346': set(['157', '441', '346', \
'445', '436', '247', '535', '252', '256']), '347': set(['347', \
'158', '441', '446', '248', '536', '2', '437', '252', '253', \
'442', '257']), '340': set(['151', '529', '245', '246', '241', \
'439', '340', '250', '435', '434', '430']), '341': set(['530', \
'152', '440', '246', '341', '242', '251', '435', '431']), '342': \
set(['153', '243', '441', '531', '436', '247', '342', '252', \
'437', '432', '248']), '343': set(['154', '442', '433', '244', \
'532', '438', '343', '253', '437', '249', '248']), '348': \
set(['159', '348', '447', '537', '443', '258', '438', '6', \
'253', '442', '249', '254'])}
```

APPENDIX F

EXAMPLE SENSOR LOCATION FILE

This file contains an appended dictionary of coordinates for each potential sensor location. This file is named `parsed_sensor_locations.py` and is needed in the `sensor_placement.py` file.

```
sens_coord =  { 344 :( 45.5 , 7.5 , 20.5 ), 345 \
:( 48.5 , 7.5 , 20.5 ), 346 :( 38 , 9 , 20.5 ), 347 \
:( 41 , 9 , 20.5 ), 340 :( 47 , 6 , 20.5 ), 341 :( \
50 , 6 , 20.5 ), 342 :( 39.5 , 7.5 , 20.5 ), 343 :( \
42.5 , 7.5 , 20.5 ), 348 :( 44 , 9 , 20.5 ), 349 :( \
47 , 9 , 20.5 ), 595 :( 42.5 , -3 , 23.5 ), 719 :( \
50 , 6 , 24.5 ), 718 :( 47 , 6 , 24.5 ), 717 :( 44 , \
6 , 24.5 ), 716 :( 41 , 6 , 24.5 ), 715 :( 38 , 6 , \
24.5 ), 714 :( 48.5 , 4.5 , 24.5 ), 713 :( 45.5 , \
4.5 , 24.5 ), 712 :( 42.5 , 4.5 , 24.5 ), 711 :( \
39.5 , 4.5 , 24.5 ), 710 :( 50 , 3 , 24.5 ), 606 :( \
48.5 , 0 , 23.5 ), 594 :( 39.5 , -3 , 23.5 ), 736 :( \
47 , 12 , 24.5 ), 619 :( 47 , 4.5 , 23.5 ), 298 :( \
42.5 , -7.5 , 20.5 )}
```

APPENDIX G

EXAMPLE TEVA-SPOT IMPACT FILE

A shortened version of the impact file used for the TEVA-SPOT chapter is included below.

```
994
1 0
121210 -1 5000 5000.0
121210 407 23 23.9
121210 4 385 385.2
121210 8 58 58.6
121210 408 24 24.4
121210 59 77 77.1
121210 58 215 215.5
121210 55 147 147.1
121210 54 141 141.8
121210 56 189 189.9
121210 50 184 184.5
121210 53 108 108.5
121210 52 303 303.4
121210 298 31 31.6
121210 299 35 35.3
121210 291 76 76.2
121210 179 100 100.0
121210 195 55 55.4
```

121210 194 38 38.3

121210 67 60 60.1

121210 191 243 243.1

121210 193 302 302.2

121210 68 68 68.3

121210 89 403 403.9

121210 113 187 187.5

121210 176 254 254.3

121210 82 93 93.2

121210 83 48 48.5

121210 81 198 198.9

121210 86 131 131.4

121210 84 42 42.9

121210 85 54 54.7

121210 7 99 99.6

121210 308 157 157.7

121210 300 158 158.9

121210 106 118 118.3

121210 104 343 343.2

121210 105 157 157.9

121210 39 331 331.2

121210 60 95 95.4

121210 61 112 112.3

121210 62 151 151.8

121210 65 156 156.3

121210 66 125 125.2

121210 178 80 80.3

121210 177 61 61.3

121210 69 88 88.7

121210 175 233 233.7

121210 172 127 127.4

121210 171 136 136.4

121210 183 160 160.2

121210 180 134 134.8

121210 186 44 44.3

121210 187 83 83.8

121210 184 100 100.5

121210 6 138 138.2

121210 188 160 160.2

121210 196 162 162.5

121210 185 47 47.0

121210 99 158 158.1

121210 98 80 80.9

121210 168 368 368.4

121210 91 58 58.5

121210 165 377 377.1

121210 160 204 204.0

121210 97 109 109.1

121210 10 85 85.9

121210 15 466 466.4

121210 14 67 67.0

121210 17 223 223.2

121210 19 207 207.4

121210 153 254 254.5

121210 155 426 426.5

121210 159 296 296.1

121210 48 197 197.8

121210 49 177 177.2

121210 46 300 300.7

121210 47 157 157.4

121210 45 370 370.5

121210 42 279 279.5

121210 43 205 205.4

121210 40 348 348.9

121210 41 240 240.5

121210 9 308 308.5

121210 201 73 73.9

121210 207 170 170.7

121210 77 74 74.9

121210 76 49 49.4

121210 75 49 49.7

121210 74 107 107.3

121210 73 135 135.1

121210 70 120 120.7

121210 78 129 129.6

151230 -1 5000 5000.0

151230 153 137 137.1

151230 276 62 62.4

151230 159 139 139.3

151230 158 383 383.6

151230 178 227 227.1

151230 499 23 23.3

151230 50 106 106.4

151230 60 132 132.7

151230 61 93 93.1

151230 62 83 83.6

151230 275 82 82.1

151230 179 149 149.4

151230 67 307 307.9

151230 68 173 173.1

151230 69 125 125.2

151230 497 69 69.2

151230 378 77 77.5

151230 172 102 102.7

151230 171 149 149.0

151230 48 120 120.2

151230 49 110 110.3

151230 47 162 162.4

151230 42 247 247.4

151230 43 132 132.5

151230 41 231 231.1

151230 599 58 58.7

151230 180 139 139.4

151230 5 139 139.8

151230 492 40 40.7

151230 7 154 154.4

151230 6 154 154.5

151230 281 59 59.9

151230 188 155 155.1

151230 505 50 50.4

151230 166 93 93.2

151230 4 80 80.2

151230 717 37 37.3

151230 613 21 21.3

151230 77 187 187.0

151230 384 81 81.2

151230 187 261 261.1

151230 70 110 110.9

151230 164 122 122.0

151230 165 79 79.0

151230 269 130 130.4

151230 160 116 116.4

151230 78 144 144.5

151230 263 107 107.4

151230 485 52 52.2

151230 55 89 89.7

151230 54 112 112.5

151230 270 101 101.9

151230 56 82 82.7